note.4.149323399sp note.4.249323399sp note.4.349323399sp note.5.169323399sp note.7.173729359sp note.11.1129323399sp note.12.1133729359sp note.20.1209323399sp note.20.2209323399sp note.21.1213729359sp note.21.2213729359sp note.21.3213729359sp note.21.4213729359sp note.21.5213729359sp note.22.1229323399sp note.25.1253729359sp note.30.1309323399sp note.30.2309323399sp note.31.1313729359sp note.31.2313729359sp note.32.1329323399sp note.32.2329323399sp note.34.1349323399sp note.34.2349323399sp note.34.3349323399sp note.40.14020542551sp note.40.24026674324sp note.41.1413729359sp note.42.1429323399sp note.43.1433729359sp note.43.2433729359sp note.43.3433729359sp note.43.4449323399sp note.44.14423856389sp note.44.2449323399sp note.49.1493729359sp note.49.2493729359sp note.50.1509323399sp note.51.1513729359sp note.51.2513729359sp note.52.1529323399sp note.53.1533729359sp note.60.1609323399sp note.60.26032310227sp note.61.16122596313sp note.62.1629323399sp note.63.1633729359sp note.64.1649323399sp note.64.2649323399sp note.69.1694888921sp note.69.2693729359sp note.70.1709323399sp note.72.1729323399sp note.72.2729323399sp note.72.3729323399sp note.74.1749323399sp note.75.1753729359sp

Block 3

Komplexität, Kausalität & Generalisierung

Vorlesungsbegleitende Notizen

Data Science and Strategy for Business

Joerg Osterrieder

31. März 2026

Diese Notizen begleiten die Vorlesung und vertiefen die Konzepte aus den Folien. Sie folgen Lena, einer Junior Data Scientist bei DataCo, durch die Herausforderungen der fortgeschrittenen Datenanalyse.

Inhaltsverzeichnis

1 Kann ich meinem Modell vertrauen? – BLUE und Gauss-Markov **4**
1.1 Entdeckungsfrage . . . 4
1.2 Die Waage-Analogie . . . 4
1.3 BLUE erklaert . . . 4
1.4 Das Gauss-Markov-Theorem . . . 6
1.5 Annahmen pruefen . . . 7
1.6 Beispiel mit Zahlen . . . 8
1.7 R-Code: Annahmen pruefen . . . 10
1.8 Uebungsaufgaben . . . 10

2 Das Zwillingsproblem – Multikollinearitaet und VIF **11**
2.1 Entdeckungsfrage . . . 11
2.2 Die Zwillings-Analogie . . . 12
2.3 Multikollinearitaet erklaert . . . 12
2.4 Der Variance Inflation Factor (VIF) . . . 13
2.5 Beispiele mit Zahlen . . . 14
2.6 Fuenf Loesungsansaetze . . . 15
2.7 R-Code: VIF berechnen . . . 18
2.8 Uebungsaufgaben . . . 18

3 Der Auswendiglerner – Bias-Varianz-Tradeoff **20**
3.1 Entdeckungsfrage . . . 20
3.2 Die Auswendiglerner-Analogie . . . 20
3.3 Overfitting und Underfitting . . . 20
3.4 In-Sample vs. Out-of-Sample . . . 21
3.5 Die Bias-Varianz-Zerlegung . . . 21
3.6 Die Zielscheiben-Analogie . . . 22
3.7 Polynomgrade im Vergleich . . . 25
3.8 Faustregel: Beobachtungen pro Praediktor . . . 25
3.9 R-Code: Overfitting demonstrieren . . . 28
3.10 Uebungsaufgaben . . . 28

4 Die Leine fuer wilde Koeffizienten – Regularisierung **30**
4.1 Entdeckungsfrage . . . 30
4.2 Die Leine und der Tuersteher . . . 30
4.3 Die Strafidee . . . 30
4.4 Ridge Regression (L2) . . . 31
4.5 Lasso Regression (L1) . . . 32
4.6 Elastic Net . . . 32
4.7 Vergleichstabelle: Ridge vs. Lasso vs. Elastic Net . . . 33
4.8 Lambda waehlen . . . 34
4.9 Beispiele mit Zahlen . . . 36
4.10 R-Code: glmnet Pipeline . . . 38
4.11 Uebungsaufgaben . . . 39

5 Rotierende Richter – Kreuzvalidierung **40**
5.1 Entdeckungsfrage . . . 40
5.2 Die Analogie der rotierenden Richter . . . 40
5.3 K-Fold Cross-Validation Schritt fuer Schritt . . . 41
5.4 CV-Varianten . . . 42

5.5 CV und Regularisierung: Das Dream-Team . . . 44
5.6 Beispiel mit Zahlen . . . 45
5.7 R-Code: caret und cv.glmnet . . . 46
5.8 Uebungsaufgaben . . . 47

6 Verursacht mein Regenschirm den Regen? – Kausalitaet 49
6.1 Entdeckungsfrage . . . 49
6.2 Regenschirm und Puppenspieler . . . 49
6.3 Korrelation vs. Kausalitaet . . . 50
6.4 Fuenf Kriterien fuer Kausalitaet . . . 50
6.5 Pearls Leiter der Kausalitaet . . . 51
6.6 Confounding erklaert . . . 52
6.7 Gerichtete azyklische Graphen (DAGs) . . . 53
6.8 Methoden der kausalen Inferenz (Ueberblick) . . . 53
6.9 Beispiel: Lenas Marketing-Puzzle . . . 55
6.10 R-Code: Korrelation und partielle Korrelation . . . 57
6.11 Uebungsaufgaben . . . 58

7 Das Experiment – A/B-Testing 60
7.1 Entdeckungsfrage . . . 60
7.2 A/B-Test Prinzipien . . . 60
7.3 Testdesign: Vor dem Experiment . . . 61
7.4 Stichprobengroesse und Power . . . 62
7.5 Cohens *d*: Effektstaerke . . . 63
7.6 Drei gefaehrliche Fehler . . . 64
7.7 Beispiel: Lenas vollstaendiger A/B-Test . . . 65
7.8 R-Code: t.test, Cohens *d*, und Power-Analyse . . . 67
7.9 Uebungsaufgaben . . . 68

8 Ja oder Nein – Logistische Regression 69
8.1 Entdeckungsfrage . . . 69
8.2 Lichtschalter vs. Dimmer . . . 69
8.3 Warum OLS bei binaeren Daten versagt . . . 69
8.4 Die logistische Funktion . . . 70
8.5 Log-Odds und Odds Ratios . . . 71
8.6 Confusion Matrix . . . 72
8.7 ROC-Kurve und AUC . . . 74
8.8 Beispiele: DataCo und Titanic . . . 77
8.9 Anwendungsgebiete . . . 78
8.10 R-Code: glm, Odds Ratios, und ROC . . . 80
8.11 Uebungsaufgaben . . . 80
8.12 Lenas Reise: Zusammenfassung . . . 81

Loesungen zu den Uebungsaufgaben 83

1 Kann ich meinem Modell vertrauen? – BLUE und Gauss-Markov

Lenas Herausforderung

Lena hat gerade bei DataCo angefangen. Ihr Chef übergibt ihr einen Datensatz mit 50.000 Kundendatensätzen und sagt: „Sag mir, wer kündigt." Lena öffnet RStudio, tippt `lm(churn ~ ., data = df)`, und bekommt sofort Koeffizienten, Standardfehler und p-Werte. Aber dann fragt sie sich: *Kann ich diesen Zahlen vertrauen? Wann sind sie zuverlässig – und wann nicht?*

1.1 Entdeckungsfrage

Entdeckungsfrage

Was macht eine gute Waage aus? Stellen Sie sich vor, Sie wiegen denselben Gegenstand zehnmal hintereinander. Was wünschen Sie sich von den Ergebnissen – und was wäre schlimm?

1.2 Die Waage-Analogie

Schätzer: Eine Rechenvorschrift, die aus Stichprobendaten einen unbekannten Parameterwert berechnet.

Bevor wir in die Mathematik einsteigen, nutzen wir ein Bild, das jeder kennt: eine Küchenwaage. Stellen Sie sich vor, Sie kaufen eine neue Waage und wollen prüfen, ob sie taugt. Sie legen ein Päckchen Butter (250 g) zehnmal hintereinander auf die Waage und notieren die Ergebnisse.

Eigenschaft 1: Genau (Unbiased). Eine gute Waage zeigt *im Mittel* den richtigen Wert an. Wenn Sie zehnmal wiegen und den Durchschnitt bilden, sollte er bei 250 g liegen – nicht systematisch bei 245 g oder 260 g. Eine Waage, die systematisch zu hoch oder zu niedrig anzeigt, ist *verzerrt* (biased). In der Statistik heißt das: Der Erwartungswert des Schätzers trifft den wahren Parameter.

Eigenschaft 2: Stabil (Low Variance). Wenn die zehn Messungen zwischen 248 g und 252 g schwanken, ist das akzeptabel. Wenn sie zwischen 200 g und 300 g schwanken, ist die Waage unbrauchbar – selbst wenn der Mittelwert zufällig bei 250 g liegt. Stabilität bedeutet geringe Streuung um den Mittelwert. In der Statistik: Der Schätzer hat eine kleine Varianz.

Eigenschaft 3: Beste (Best). Nun stellen Sie sich vor, es gäbe viele verschiedene Waagen, die alle genau sind (alle im Mittel 250 g anzeigen). Welche wählen Sie? Natürlich die stabilste – also diejenige mit der geringsten Streuung. „Best" heißt in der Statistik: unter allen genau-messenden Waagen die präziseste.

Diese drei Eigenschaften zusammen ergeben das Akronym BLUE, das wir im nächsten Abschnitt formal definieren.

1.3 BLUE erklärt

BLUE: Best Linear Unbiased Estimator – der beste lineare erwartungstreue Schätzer.

Das Akronym BLUE steht für **B**est **L**inear **U**nbiased **E**stimator. Jeder Buchstabe hat eine präzise Bedeutung, die wir nun einzeln aufschlüsseln:

OLS: Ordinary Least Squares – Methode der kleinsten Quadrate

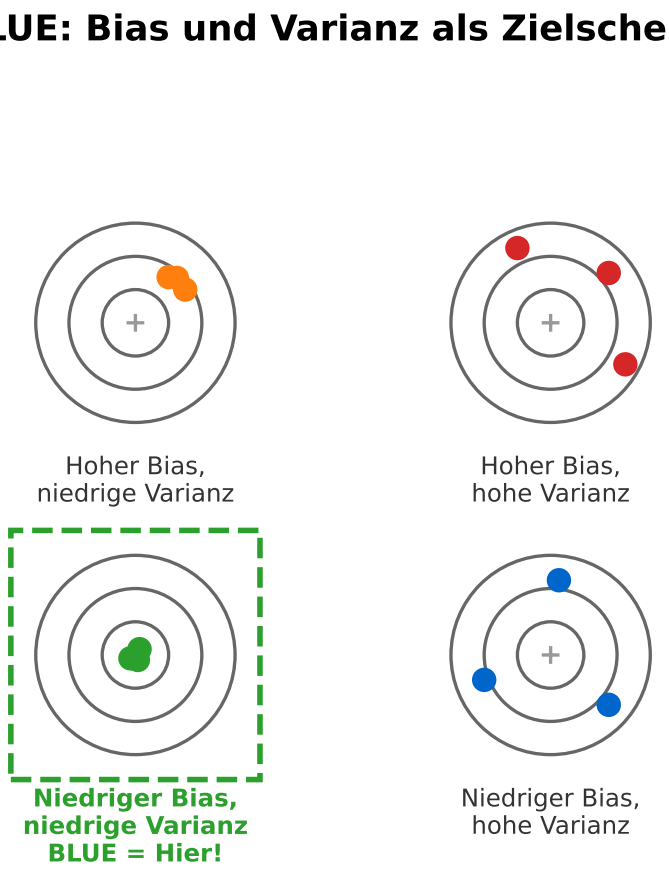


Abbildung 1: Zielscheiben-Analogie: Genauigkeit und Präzision eines Schätzers

BLUE – Die vier Buchstaben

E – Estimator (Schätzer): Eine mathematische Rechenvorschrift, die aus Daten einen Schätzwert berechnet. Der OLS-Schätzer $\hat{\beta}$ ist ein Beispiel: Er nimmt die Daten (X, y) und liefert geschätzte Koeffizienten.
U – Unbiased (Erwartungstreu): Stellen Sie sich vor, Lena könnte ihr Experiment 1000-mal wiederholen: jedes Mal 50.000 neue Kunden ziehen und $\hat{\beta}$ berechnen. Wenn der Mittelwert aller 1000 Schätzungen den *wahren* Koeffizienten β trifft, ist der Schätzer erwartungstreu. Formal: $E[\hat{\beta}] = \beta$.
L – Linear: Der Schätzer ist eine *lineare Funktion der beobachteten y-Werte*. Das bedeutet, man kann $\hat{\beta}$ schreiben als gewichtete Summe der Beobachtungen: $\hat{\beta} = \sum_i w_i \cdot y_i$, wobei die Gewichte w_i nur von den x-Werten abhängen, nicht von y selbst.
B – Best (Bester): Unter *allen* Schätzern, die gleichzeitig linear und erwartungstreu sind, hat OLS die **kleinste Varianz**. Es gibt keinen anderen linearen, erwartungstreuen Schätzer, der präziser ist.

Was bedeutet das für Lena? Wenn bestimmte Bedingungen erfüllt sind (dazu gleich mehr), dann ist ihr `lm()`-Output tatsächlich das Beste, was sie mit einem linearen Modell erreichen kann. Kein anderer linearer Schätzer wäre präziser. Aber Vorsicht: „Best" bezieht sich nur auf Varianz, nicht auf den Gesamtfehler. Ein Schätzer mit etwas Verzerrung, aber deutlich weniger Varianz, kann einen niedrigeren Gesamtfehler (MSE) haben – das ist die Grundidee der Regularisierung, die wir in Abschnitt 4 kennenlernen.

1.4 Das Gauss-Markov-Theorem

Gauss-Markov-Theorem: Unter den 5 Standardannahmen ist OLS BLUE.

Die BLUE-Eigenschaft fällt nicht vom Himmel. Sie gilt nur, wenn fünf Annahmen erfüllt sind. Diese Annahmen bilden das **Gauss-Markov-Theorem**, eines der zentralen Resultate der klassischen Statistik.

Die fünf Gauss-Markov-Annahmen

Annahme 1: Linearität in den Parametern.

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip} + \varepsilon_i$$

Dabei ist y_i die Zielgröße für Beobachtung i, die β_j sind die unbekannten Parameter, x_{ij} ist der Wert des j-ten Prädiktors für Beobachtung i, und ε_i ist der Fehlerterm. Beachten Sie: Das Modell muss linear in den *Parametern* β sein, nicht notwendig in den x-Variablen. Zum Beispiel ist $y = \beta_0 + \beta_1 x^2 + \varepsilon$ linear in β, obwohl x quadriert eingeht.

Annahme 2: Zufällige Stichprobe. Die Beobachtungen (x_i, y_i) sind unabhängig voneinander und stammen aus derselben Grundgesamtheit. Bei Lenas 50.000 Kunden bedeutet das: Kein Kunde beeinflusst die Daten eines anderen, und alle Kunden kommen aus demselben Markt.

Annahme 3: Keine perfekte Multikollinearität. Kein Prädiktor darf eine exakte Linearkombination der anderen sein. Also z. B. nicht $x_3 = 2x_1 + x_2$. Approximative Korrelation (wie $r = 0{,}9$) ist erlaubt, führt aber zu großen Standardfehlern – das ist das Thema von Abschnitt 2.

Annahme 4: Exogenität (Erwartungswert der Fehler ist Null).

$$E[\varepsilon_i \mid X] = 0$$

Der Fehlerterm ε_i ist im Mittel Null, gegeben alle x-Werte. Das bedeutet: Es gibt keinen systematischen Zusammenhang zwischen den Prädiktoren und dem Fehler. Wenn diese Annahme verletzt ist (z. B. weil eine wichtige Variable fehlt), sind die Schätzer *verzerrt*.

Annahme 5: Homoskedastizität (Konstante Fehlervarianz).

$$\mathrm{Var}(\varepsilon_i \mid X) = \sigma^2 \quad \text{für alle } i$$

Die Streuung der Fehler ist für alle Beobachtungen gleich. Bei Lenas Daten: Die Vorhersageunsicherheit sollte für Kunden mit geringer Nutzung genauso groß sein wie für Power-User. Wenn die Streuung mit der Nutzung wächst („Trichter-Muster"), liegt Heteroskedastizität vor.

Was das Theorem sagt: Wenn alle fünf Annahmen erfüllt sind, dann ist der OLS-Schätzer BLUE. Es gibt dann keinen anderen linearen, erwartungstreuen Schätzer mit kleinerer Varianz.

Ein häufiges Missverständnis: Viele Studierende denken, dass auch Normalverteilung der Fehler zu den Gauss-Markov-Annahmen gehört. Das ist *nicht* der Fall. Normalität wird erst benötigt, wenn wir t-Tests und Konfidenzintervalle berechnen wollen – also für die *Inferenz*. Für die BLUE-Eigenschaft allein reichen die fünf genannten Annahmen.

1.5 Annahmen prüfen

In der Praxis sind die Gauss-Markov-Annahmen selten perfekt erfüllt. Deshalb ist es entscheidend, sie systematisch zu prüfen. Die folgende Tabelle zeigt für jede Annahme die passende Diagnosemethode und den zugehörigen R-Befehl:

Annahme	Diagnostik	R-Funktion
Linearität	Residuen vs. Fitted Plot	`plot(model, 1)`
Zufällige Stichprobe	Studiendesign prüfen	(theoretisch)
Keine perf. Multikoll.	VIF berechnen	`car::vif(model)`
$E[\varepsilon\|X] = 0$	Theorie & Residuenanalyse	`plot(model, 1)`
Homoskedastizität	Scale-Location Plot, Breusch-Pagan	`lmtest::bptest()`
Zusätzlich:		
Normalität	Q-Q Plot	`shapiro.test(resid())`
Keine Autokorrelation	Durbin-Watson Test	`lmtest::dwtest()`

Anscombes Quartett: Vier Datensätze mit identischen Statistiken, aber völlig unterschiedlichen Mustern.

Ein eindrucksvolles Beispiel dafür, warum Visualisierung unverzichtbar ist, liefert **Anscombes Quartett** (1973). Francis Anscombe konstruierte vier Datensätze, die alle denselben Mittelwert ($\bar{x} = 9$, $\bar{y} = 7{,}5$), dieselbe Standardabweichung, dieselbe Korrelation ($r = 0{,}82$) und dieselbe Regressionslinie haben. Trotzdem zeigen die vier Datensätze völlig verschiedene Muster: einer ist linear, einer kurvilinear, einer hat einen Ausreißer, und einer hat eine einzige einflussreiche Beobachtung. Die Lektion ist klar: Kennzahlen allein reichen nicht – man muss die Daten *sehen*.

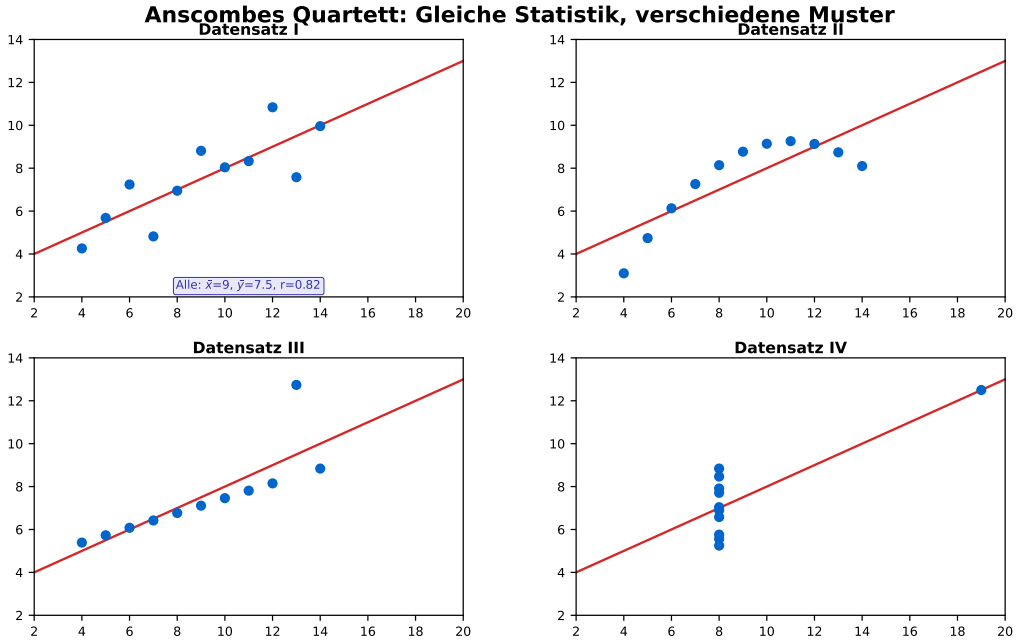


Abbildung 2: Anscombes Quartett: Vier Datensätze, eine Statistik, vier verschiedene Geschichten.

Für Lena bei DataCo bedeutet das: Bevor sie sich auf Koeffizienten und p-Werte verlässt, sollte sie ihre Residuen plotten. Der Befehl `plot(model)` in R erzeugt automatisch vier Diagnose-Grafiken, die die wichtigsten Annahmen visuell prüfen.

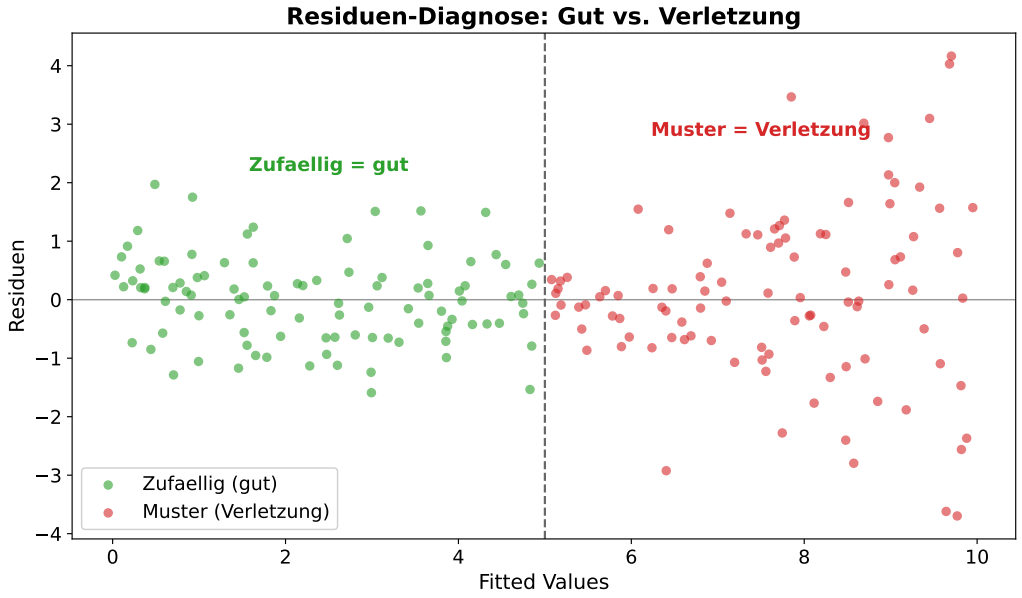


Abbildung 3: Auswirkungen von Annahmenverletzungen auf Residuenplots

1.6 Beispiel mit Zahlen

Korrekte vs. verletzte Annahmen – Was ändert sich?

Lena schätzt bei DataCo ein Regressionsmodell mit $n = 500$ Beobachtungen und $p = 4$ Prädiktoren. Sie betrachtet den Koeffizienten $\hat{\beta}_1$ (Einfluss der Vertragslaufzeit auf Churn) unter zwei Szenarien:

Szenario	$\hat{\beta}_1$	SE	t	p
Korrekte Annahmen	2,3	0,4	5,75	$< 0{,}001$
Mit Heteroskedastizität	2,3	0,8	2,88	0,004

Was fällt auf? Der geschätzte Koeffizient $\hat{\beta}_1 = 2{,}3$ bleibt in beiden Szenarien gleich – OLS ist immer noch erwartungstreu, auch bei Heteroskedastizität. Aber der Standardfehler *verdoppelt* sich von 0,4 auf 0,8. Dadurch *halbiert* sich der t-Wert (von 5,75 auf 2,88), und der p-Wert steigt um den Faktor 4.

In diesem Beispiel bleibt das Ergebnis signifikant. Aber stellen Sie sich vor, der wahre Effekt wäre kleiner: Bei einem Koeffizienten von $\hat{\beta}_1 = 1{,}0$ würde der t-Wert unter Heteroskedastizität auf $1{,}0/0{,}8 = 1{,}25$ fallen – weit unter dem kritischen Wert von 1,96. Die *Signifikanz verschwindet*, obwohl der Effekt real ist.

Fazit: Verletzte Annahmen ändern nicht die Punktschätzung, aber sie blähen die Standardfehler auf und schwächen die statistische Inferenz. Im schlimmsten Fall übersieht Lena einen echten Effekt, weil der p-Wert zu groß geworden ist.

Gauss und der Asteroid Ceres (1801)

Im Januar 1801 entdeckte der italienische Astronom Giuseppe Piazzi einen neuen Himmelskörper – Ceres, den ersten bekannten Asteroiden. Er konnte ihn jedoch nur 41 Tage lang beobachten, bevor Ceres hinter der Sonne verschwand. Die Astronomen standen vor einem Problem: Wie findet man Ceres wieder, wenn man nur 22 Messungen über einen winzigen Bahnbogen von 3 Grad hat?
Carl Friedrich Gauss, damals 24 Jahre alt, entwickelte eine neue Methode: die **Methode der kleinsten Quadrate**. Er minimierte die Summe der quadrierten Abweichungen zwischen beobachteten und vorhergesagten Positionen. Seine Vorhersage war so präzise, dass der Astronom Franz Xaver von Zach Ceres am 31. Dezember 1801 genau dort fand, wo Gauss es vorhergesagt hatte.
Was Gauss mit 22 Messwerten und Papier tat, tut Lena bei DataCo mit 50.000 Zeilen und `lm()`. Die Methode ist dieselbe – nur die Skalierung hat sich geändert. Rund 100 Jahre später formalisierte der russische Mathematiker Andrei Markov die optimalen Eigenschaften von Gauss' Methode. Das Ergebnis kennen wir heute als das Gauss-Markov-Theorem.

Drei häufige Fehler über BLUE

Fehler 1: „BLUE heißt, OLS ist immer optimal."
Nein! BLUE gilt nur unter den fünf Gauss-Markov-Annahmen. Wenn eine Annahme verletzt ist – z. B. Heteroskedastizität – ist OLS nicht mehr „Best". Außerdem gibt es nichtlineare Schätzer (wie Ridge oder Lasso), die einen niedrigeren Gesamtfehler (MSE) haben können, indem sie etwas Verzerrung in Kauf nehmen.
Fehler 2: „Normalverteilung der Fehler ist für BLUE nötig."
Nein! Die fünf Gauss-Markov-Annahmen enthalten keine Normalitätsannahme. Normalität brauchen wir erst für Inferenz: t-Tests, F-Tests und Konfidenzintervalle setzen normalverteilte Fehler voraus (bei kleinem n). Bei großem n hilft der Zentrale Grenzwertsatz.
Fehler 3: „Bei verletzten Annahmen ist OLS nutzlos."
Nein! Wenn z. B. nur die Homoskedastizität verletzt ist, bleibt OLS erwartungstreu – die Koeffizienten sind im Mittel korrekt. Was sich ändert, sind die Standardfehler: Sie werden unzuverlässig. In diesem Fall kann man *robuste Standardfehler* verwenden (z. B. mit dem R-Paket `sandwich`), um korrekte Inferenz zu erhalten.

1.7 R-Code: Annahmen prüfen

Modelldiagnostik in R

```r
# Daten laden und Modell schaetzen
df <- read.csv("dataco_kunden.csv")
model <- lm(churn_score ~ alter + vertragslaufzeit +
            monatl_kosten + nutzung_gb, data = df)
summary(model)

# --- Diagnose-Plots (4 auf einmal) ---
par(mfrow = c(2, 2))
plot(model)

# --- Breusch-Pagan Test auf Heteroskedastizitaet ---
library(lmtest)
bptest(model)
# p < 0.05 -> Heteroskedastizitaet vorhanden

# --- Shapiro-Wilk Test auf Normalitaet ---
shapiro.test(residuals(model))
# Achtung: bei n > 5000 fast immer signifikant!

# --- VIF auf Multikollinearitaet ---
library(car)
vif(model)
# VIF > 5: problematisch, VIF > 10: schwer problematisch

# --- Robuste Standardfehler (bei Heteroskedastizitaet) ---
library(sandwich)
library(lmtest)
coeftest(model, vcov = vcovHC(model, type = "HC1"))
```

1.8 Übungsaufgaben

Aufgabe 1.1 – Waage und Schätzer

Eine Laborwaage zeigt bei 10 Messungen desselben Gewichts (wahres Gewicht: 100 g) die Werte 98, 102, 99, 101, 100, 103, 97, 101, 99, 100 an.
(a) Berechnen Sie den Mittelwert. Ist die Waage erwartungstreu?
(b) Berechnen Sie die Standardabweichung. Wie beurteilen Sie die Stabilität?
(c) Welcher BLUE-Eigenschaft entspricht (a), welcher (b)?

Aufgabe 1.2 – Gauss-Markov-Annahmen identifizieren

Für jedes Szenario: Welche Gauss-Markov-Annahme ist verletzt?
(a) Lena modelliert Umsatz als Funktion von Werbeausgaben, aber der Zusammenhang ist exponentiell.
(b) In einer Zeitreihe heutiger Umsatz hängt vom gestrigen ab.
(c) Lena nimmt „Monatliche Kosten" und „Jahreskosten = 12 × Monatliche Kosten" auf.
(d) Die Streuung der Residuen nimmt mit steigendem Umsatz zu.

Aufgabe 1.3 – Heteroskedastizität und Standardfehler

In einem Modell mit $n = 200$ ergibt sich $\hat{\beta}_1 = 3{,}5$ mit $SE = 0{,}7$ unter korrekten Annahmen.
(a) Berechnen Sie den t-Wert und prüfen Sie die Signifikanz ($\alpha = 0{,}05$).
(b) Durch Heteroskedastizität verdreifacht sich der SE auf 2,1. Wie ändert sich der t-Wert?
(c) Ist $\hat{\beta}_1$ immer noch signifikant? Was bedeutet das für Lenas Interpretation?

Aufgabe 1.4 – R-Diagnoseplot interpretieren

Lena führt `plot(model)` aus und sieht im Residuen-vs-Fitted-Plot ein deutliches U-förmiges Muster.
(a) Welche Annahme ist verletzt?
(b) Was bedeutet das für die Koeffizientenschätzungen?
(c) Nennen Sie zwei mögliche Lösungen.

Aufgabe 1.5 – BLUE oder nicht BLUE?

Entscheiden Sie für jeden Schätzer, ob er die BLUE-Eigenschaft hat, und begründen Sie:
(a) $\hat{\beta}_1^{\text{OLS}}$ bei erfüllten Gauss-Markov-Annahmen.
(b) Der Median als Schätzer für den Erwartungswert.
(c) $\hat{\beta}_1^{\text{Ridge}}$ (Ridge-Regression mit $\lambda > 0$).

Kernaussage: OLS-Schätzer sind BLUE – aber *nur* unter den 5 Gauss-Markov-Annahmen. In der Praxis muss jede Annahme geprüft werden: mit Residuenplots, VIF und Breusch-Pagan-Test. Bei Verletzungen ändern sich nicht die Koeffizienten, sondern die Standardfehler – und damit die gesamte Inferenz.

2 Das Zwillingsproblem – Multikollinearität und VIF

Lenas Herausforderung

Lena ist zufrieden mit ihrem ersten Modell – die Annahmen hat sie geprüft. Jetzt will sie es verbessern und fügt zwei neue Variablen hinzu: „Vertragslaufzeit" und „monatliche Kosten". Aber etwas Seltsames passiert: Der Koeffizient für Vertragslaufzeit wechselt das Vorzeichen (er war positiv, jetzt ist er negativ), die Standardfehler explodieren, und ein vorher signifikanter Prädiktor verliert plötzlich seine Signifikanz. Was ist passiert?

2.1 Entdeckungsfrage

Entdeckungsfrage

Was passiert, wenn wir eine Variable exakt kopieren und beide Kopien ins Modell stecken? Kann das Modell die einzelnen Effekte noch trennen? Und was ändert sich, wenn die Kopie nicht perfekt ist, sondern nur *fast* identisch?

2.2 Die Zwillings-Analogie

Multikollinearität: Starke lineare Korrelation zwischen zwei oder mehr Prädiktoren in einem Regressionsmodell.

Stellen Sie sich eine Gerichtsverhandlung vor. Zwei Zeugen sollen aussagen, was geschehen ist. Wenn die beiden Zeugen völlig unabhängig voneinander berichten, kann der Richter aus beiden Aussagen unterschiedliche Informationen gewinnen: Der eine beschreibt vielleicht den Täter, der andere den Fluchtweg. Jeder trägt etwas Eigenes bei.

Jetzt stellen Sie sich vor, die beiden Zeugen sind eineiige Zwillinge, die immer zusammen waren und immer dasselbe gesehen haben. Ihre Aussagen sind praktisch identisch. Der Richter hört zweimal dasselbe – aber kann nicht unterscheiden, welcher Zwilling welche Information beiträgt. Wenn er fragt „Was genau haben *Sie* gesehen?", zeigt jeder Zwilling auf den anderen und sagt: „Dasselbe wie er."

Genau das passiert bei Multikollinearität. Wenn zwei Prädiktoren (X_1 und X_2) hoch korreliert sind, tragen sie fast dieselbe Information ins Modell. OLS kann nicht zuverlässig aufteilen, welcher Anteil des Effekts auf X_1 und welcher auf X_2 zurückgeht. Das Ergebnis: Die Koeffizienten werden instabil, die Standardfehler groß, und kleine Änderungen in den Daten können große Änderungen in den Schätzwerten verursachen.

Wichtig: Das Gesamtmodell kann trotzdem gut vorhersagen – die *gemeinsame* Information der Zwillinge ist ja vorhanden. Das Problem liegt in der *Interpretation* der einzelnen Koeffizienten.

2.3 Multikollinearität erklärt

Multikollinearität – Exakt vs. Approximativ

Exakte (perfekte) Multikollinearität: Ein Prädiktor ist eine *exakte* Linearkombination der anderen. Beispiel: $x_3 = 2x_1 + x_2$. In diesem Fall kann OLS die Koeffizienten gar nicht berechnen – die Formel bricht zusammen. R gibt eine Fehlermeldung oder setzt einen Koeffizienten auf `NA`.

Approximative (imperfekte) Multikollinearität: Prädiktoren sind *stark*, aber nicht perfekt korreliert. Beispiel: Korrelation $r = 0{,}92$ zwischen Vertragslaufzeit und monatlichen Kosten. OLS funktioniert technisch, aber die Schätzungen werden unzuverlässig. Dies ist der häufigere und tückischere Fall in der Praxis.

Multikollinearität zeigt sich durch vier charakteristische Symptome, die Lena bei ihrem erweiterten Modell beobachtet:

Symptom 1: Hohes R^2, aber keine signifikanten Koeffizienten. Das Gesamtmodell erklärt viel Varianz, aber kein einzelner Prädiktor erreicht statistische Signifikanz. Das klingt paradox: Wie kann das Modell als Ganzes gut sein, aber kein einzelner Prädiktor? Die Antwort: Die Prädiktoren teilen sich die Erklärungskraft, und keiner bekommt genug zugeteilt, um für sich allein signifikant zu sein.

Symptom 2: Aufgeblähte Standardfehler. Die SE der Koeffizienten sind unverhältnismäßig groß. Dadurch werden die t-Werte klein und die p-Werte groß. Lena findet keinen signifikanten Effekt, obwohl er in Wahrheit existiert.

Symptom 3: Instabile Koeffizienten. Wenn Lena eine einzige Beobachtung entfernt oder hinzufügt, ändern sich die Koeffizienten dramatisch. Ein stabiles Modell sollte robust gegenüber kleinen Datenänderungen sein.

Symptom 4: Vorzeichenwechsel. Ein Prädiktor, der theoretisch einen positiven Effekt haben sollte, bekommt plötzlich ein negatives Vorzeichen. Das liegt daran, dass OLS versucht, überlappende Information aufzuteilen, und dabei zu „überkompensierten" Schätzungen kommt.

2.4 Der Variance Inflation Factor (VIF)

VIF: Variance Inflation Factor – misst, wie stark die Varianz eines Koeffizienten durch Korrelation mit anderen Prädiktoren aufgeblasen wird.

Um Multikollinearität zu *quantifizieren*, brauchen wir ein Maß. Die einfache Korrelationsmatrix zeigt nur paarweise Zusammenhänge, übersieht aber komplexere Muster (z. B. wenn X_3 eine Linearkombination von X_1 und X_2 ist, ohne mit einem von beiden hoch korreliert zu sein). Der **Variance Inflation Factor (VIF)** löst dieses Problem.

Die Idee: Wir regressieren jeden Prädiktor X_j auf *alle* anderen Prädiktoren und berechnen das R_j^2 dieser Hilfsregression. Wenn R_j^2 hoch ist, wird X_j gut durch die anderen erklärt – also liegt Multikollinearität vor.

Der Variance Inflation Factor

$$VIF_j = \frac{1}{1 - R_j^2}$$

Symbol für Symbol:

- VIF_j – der Variance Inflation Factor für den j-ten Prädiktor
- R_j^2 – das Bestimmtheitsmaß (R^2) der Regression von X_j auf alle übrigen Prädiktoren $X_1, \ldots, X_{j-1}, X_{j+1}, \ldots, X_p$
- Wenn $R_j^2 = 0$ (keine Korrelation mit anderen Prädiktoren): $VIF_j = 1$ – keine Inflation
- Wenn $R_j^2 = 0{,}5$: $VIF_j = 2$ – Varianz doppelt so groß
- Wenn $R_j^2 = 0{,}9$: $VIF_j = 10$ – Varianz zehnfach aufgeblasen
- Wenn $R_j^2 \to 1$: $VIF_j \to \infty$ – perfekte Multikollinearität

Faustregeln:

- $VIF < 5$: unkritisch
- $5 \leq VIF < 10$: problematisch – genauer untersuchen
- $VIF \geq 10$: schwer problematisch – Handlung erforderlich

Der Name „Variance *Inflation* Factor" sagt genau, was gemessen wird: Um welchen Faktor die Varianz des Koeffizienten $\hat{\beta}_j$ aufgeblasen wird, *im Vergleich zu einer Situation ohne Multikollinearität*. Ein VIF von 10 bedeutet: Der Standardfehler von $\hat{\beta}_j$ ist $\sqrt{10} \approx 3{,}16$-mal so groß wie er wäre, wenn X_j mit keinem anderen Prädiktor korreliert wäre.

2.5 Beispiele mit Zahlen

Beispiel 1: Körpergröße und Gewicht als Prädiktoren für Blutdruck

In einer Gesundheitsstudie mit $n = 300$ Personen werden Körpergröße und Gewicht als Prädiktoren für den systolischen Blutdruck verwendet. Größe und Gewicht korrelieren mit $r = 0{,}78$.

Modell	**Variable**	$\hat{\beta}$	SE	p
Nur Größe	Größe	0,45	0,12	$< 0{,}001$
Nur Gewicht	Gewicht	0,52	0,11	$< 0{,}001$
Beide zusammen	Größe	0,18	0,28	0,52
Beide zusammen	Gewicht	0,25	0,26	0,34

Einzeln sind beide Prädiktoren hochsignifikant. Zusammen wird keiner mehr signifikant – die Standardfehler haben sich mehr als verdoppelt! Das R^2 des Gesamtmodells bleibt hoch (0,45), aber die einzelnen Koeffizienten sind unbrauchbar geworden. Die Schlussfolgerung „weder Größe noch Gewicht beeinflussen Blutdruck" wäre falsch – das Problem liegt an der Multikollinearität, nicht an fehlenden Effekten.

Beispiel 2: Lenas DataCo-Modell mit 4 Features

Lena berechnet die VIF-Werte für ihre vier Prädiktoren:

Feature	R_j^2	VIF	$\sqrt{VIF}$	Bewertung
Vertragslaufzeit	0,82	5,6	2,4	problematisch
Monatl. Kosten	0,78	4,5	2,1	grenzwertig
Alter	0,12	1,14	1,07	unkritisch
Nutzung (GB)	0,25	1,33	1,15	unkritisch

Die Spalte $\sqrt{VIF}$ zeigt den Faktor, um den der Standardfehler aufgeblasen wird. Für Vertragslaufzeit ist der SE 2,4-mal größer als nötig. Die Berechnung für Vertragslaufzeit: $VIF = \frac{1}{1-0{,}82} = \frac{1}{0{,}18} = 5{,}6$.
Fazit: Vertragslaufzeit und monatliche Kosten überlappen stark – sie messen ähnliche Kundeninformation. Alter und Nutzung sind unproblematisch.

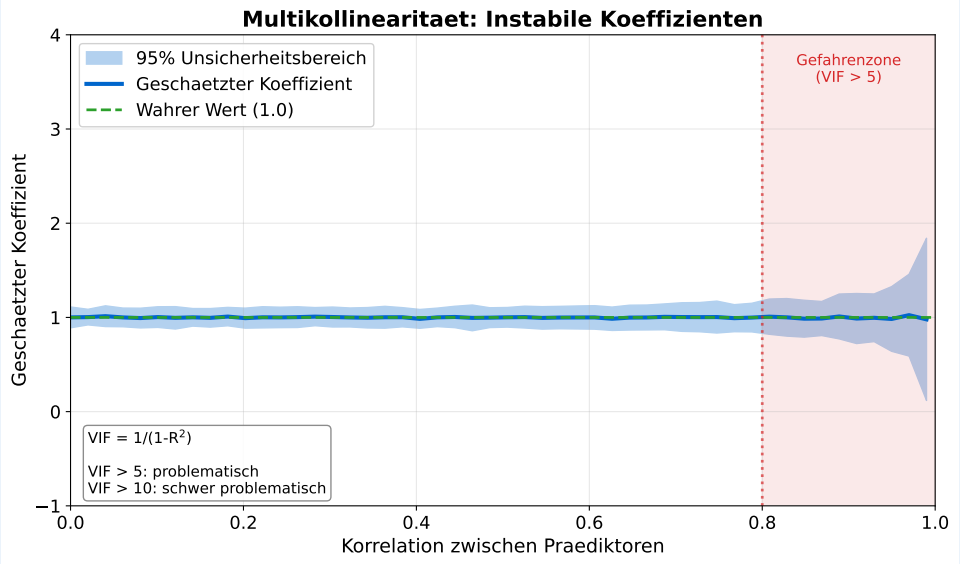


Abbildung 4: Korrelationsstruktur in Lenas DataCo-Daten.

2.6 Fünf Lösungsansätze

Wenn Lena Multikollinearität in ihrem Modell entdeckt, stehen ihr fünf Strategien zur Verfügung – von einfach bis fortgeschritten:

Lösung 1: Variable entfernen. Die einfachste Methode: Einen der korrelierten Prädiktoren aus dem Modell nehmen. Lena könnte z. B. „monatliche Kosten" entfernen und nur „Vertragslaufzeit" behalten. Vorteil: Einfach und effektiv. Nachteil: Man verliert Information, und die Entscheidung, *welche* Variable entfernt wird, sollte auf inhaltlichen Gründen basieren, nicht nur auf dem VIF-Wert. Wenn „monatliche Kosten" kausal relevanter ist als „Vertragslaufzeit", wäre es falsch, die kausale Variable zu entfernen.

Lösung 2: Variablen kombinieren. Statt eine Variable zu entfernen, kann Lena beide zu einem Index zusammenfassen. Zum Beispiel: Vertragswert = Vertragslaufzeit × monatliche Kosten. Oder sie standardisiert beide Variablen und bildet den Mittelwert. Vorteil: Keine Information geht verloren. Nachteil: Der neue Index ist schwieriger zu interpretieren.

Lösung 3: Regularisierung (Ridge / Lasso). Ridge-Regression schrumpft die Koeffizienten korrelierter Variablen zueinander, statt sie extrem schwanken zu lassen. Lasso kann korrelierte

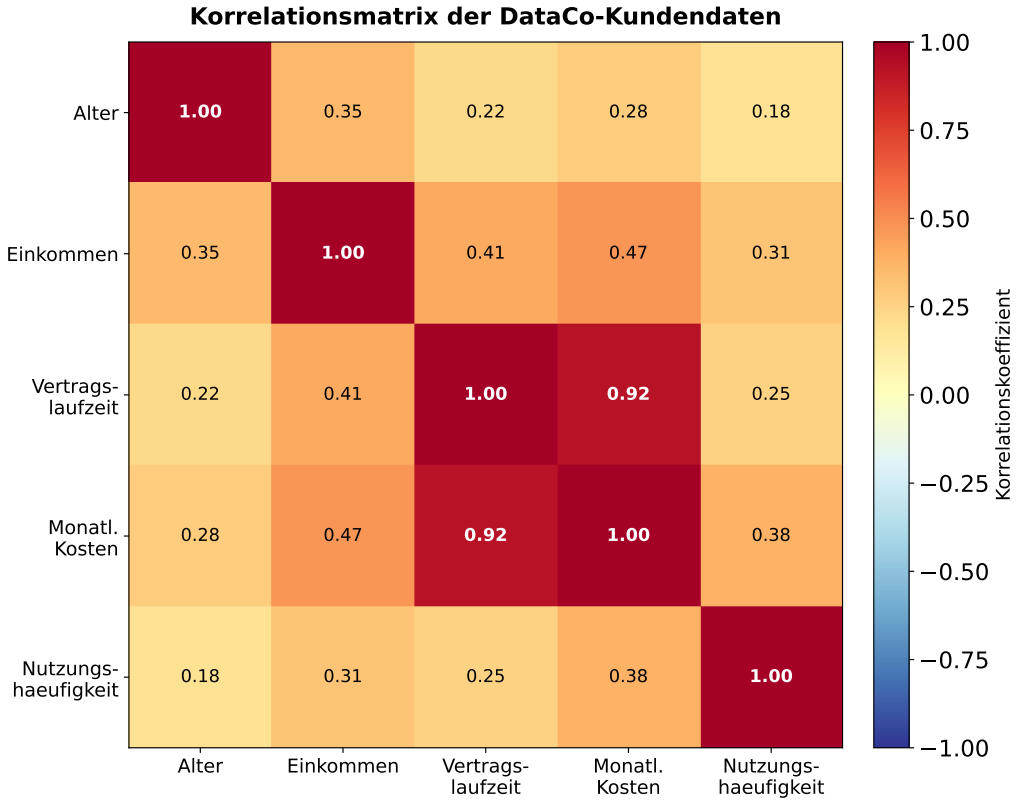


Abbildung 5: Korrelationsmatrix der DataCo-Kundendaten als Heatmap

Variablen automatisch auf Null setzen. Wir behandeln Regularisierung ausführlich in Abschnitt 4.

Lösung 4: Hauptkomponentenanalyse (PCA). PCA erzeugt neue, per Definition unkorrelierte Variablen (Hauptkomponenten) als Linearkombinationen der ursprünglichen Prädiktoren. Vorteil: Multikollinearität wird vollständig beseitigt. Nachteil: Die neuen Variablen sind abstrakt und schwer zu interpretieren.

Lösung 5: Mehr Daten sammeln. Mit mehr Beobachtungen werden die Standardfehler kleiner, und die Auswirkungen der Multikollinearität werden geringer. Dieser Ansatz ist oft nicht praktikabel, aber wenn möglich, der sauberste.

Die wichtigste Unterscheidung: Welche Lösung die richtige ist, hängt vom *Ziel* ab. Wenn Lena nur vorhersagen will, welche Kunden kündigen, ist Multikollinearität oft kein Problem – die Vorhersagekraft des Gesamtmodells leidet kaum. Wenn sie aber ihrem Chef erklären muss, *warum* Kunden kündigen (also einzelne Koeffizienten interpretieren will), muss sie die Multikollinearität behandeln.

Hoerl & Kennard: Die Erfindung von Ridge (1970)

Arthur Hoerl und Robert Kennard arbeiteten als Chemiker bei der Firma DuPont. In der industriellen Chemie ist Multikollinearität allgegenwärtig: Temperatur, Druck und Konzentration hängen oft physikalisch zusammen. Hoerl und Kennard stellten fest, dass OLS-Koeffizienten in solchen Situationen wild schwankten und absurde Werte annahmen. Ihre Idee von 1970 war einfach und elegant: Man addiert einen kleinen Strafterm λ zur Diagonale der Datenmatrix. Das „drückt" die Koeffizienten Richtung Null und stabilisiert sie. Sie nannten die Methode **Ridge Regression**, weil die Addition von λ an die Diagonale einem „Grat" (ridge) auf der Fehleroberfläche entspricht.
Damals wurde die Methode in der Statistik-Community skeptisch aufgenommen: Man opferte bewusst Erwartungstreue (BLUE wird verletzt!), um Stabilität zu gewinnen. Heute ist Ridge die Standardmethode bei Multikollinearität und die Grundlage moderner Regularisierung.

Zwei häufige Fehler über Multikollinearität

Fehler 1: „Hohe Korrelation bedeutet immer Multikollinearität."
Nicht unbedingt. Ob Multikollinearität ein *Problem* ist, hängt von drei Faktoren ab: der Höhe der Korrelation, der Stichprobengröße und dem Ziel der Analyse. Bei $n = 100.000$ kann selbst eine Korrelation von $r = 0{,}85$ tolerierbar sein, weil die großen Datenmengen die Standardfehler trotzdem klein halten. Bei $n = 50$ wäre dieselbe Korrelation katastrophal. Der VIF gibt ein besseres Bild als die bloße Korrelation.
Fehler 2: „Einfach die Variable mit dem höchsten VIF entfernen."
Gefährlich! Die Variable mit dem höchsten VIF kann die kausal relevanteste sein. Beispiel: Wenn „monatliche Kosten" kausal auf Churn wirkt und „Vertragslaufzeit" nur ein Proxy ist, wäre es falsch, die Kosten-Variable zu entfernen. Die Entscheidung muss auf inhaltlichem Wissen und der kausalen Struktur basieren (siehe Abschnitt 6), nicht nur auf dem VIF-Wert.

2.7 R-Code: VIF berechnen

VIF-Analyse und Korrelationsplot in R

```r
# Modell schaetzen
model <- lm(churn_score ~ alter + vertragslaufzeit +
            monatl_kosten + nutzung_gb, data = df)

# --- VIF berechnen ---
library(car)
vif_werte <- vif(model)
print(vif_werte)
#       alter vertragslaufzeit    monatl_kosten      nutzung_gb
#        1.14             5.60             4.50            1.33

# --- Korrelationsmatrix visualisieren ---
library(corrplot)
cor_matrix <- cor(df[, c("alter", "vertragslaufzeit",
                         "monatl_kosten", "nutzung_gb")])
corrplot(cor_matrix, method = "color", type = "upper",
         addCoef.col = "black", tl.col = "black",
         title = "Korrelationsmatrix der Praediktoren")

# --- Modell ohne redundante Variable ---
model_reduziert <- lm(churn_score ~ alter +
                      vertragslaufzeit + nutzung_gb, data = df)
vif(model_reduziert)
# Alle VIF-Werte jetzt unter 2

# --- Alternative: Index bilden ---
df$vertragswert <- scale(df$vertragslaufzeit) +
                   scale(df$monatl_kosten)
model_index <- lm(churn_score ~ alter + vertragswert +
                  nutzung_gb, data = df)
summary(model_index)
```

2.8 Übungsaufgaben

Aufgabe 2.1 – VIF von Hand berechnen

Ein Modell hat drei Prädiktoren. Die Hilfsregression von X_1 auf X_2 und X_3 ergibt $R_1^2 = 0{,}75$.
(a) Berechnen Sie VIF_1.
(b) Um welchen Faktor ist der Standardfehler von $\hat{\beta}_1$ aufgeblasen?
(c) Ist dies nach den Faustregeln problematisch?

Aufgabe 2.2 – Symptome erkennen

Lena schätzt zwei Modelle:
Modell A: `churn ~ alter + nutzung_gb` → $R^2 = 0{,}35$, beide Koeffizienten signifikant.
Modell B: `churn ~ alter + nutzung_gb + vertragslaufzeit + monatl_kosten` → $R^2 = 0{,}38$, *kein* Koeffizient signifikant.
(a) Welches Symptom der Multikollinearität zeigt sich hier?
(b) Warum steigt R^2 trotzdem?
(c) Was sollte Lena als nächstes berechnen?

Aufgabe 2.3 – Lösungsstrategie wählen

Lena hat ein Churn-Modell mit den Prädiktoren Alter, Vertragslaufzeit, monatliche Kosten und Nutzung (GB). Die VIF-Analyse zeigt: Vertragslaufzeit und monatliche Kosten haben VIF-Werte von 5,6 und 4,5.
(a) Lenas Chef will wissen, welcher *einzelne* Faktor Churn am stärksten beeinflusst. Welche Lösung empfehlen Sie?
(b) Lenas Kollegin will nur vorhersagen, wer kündigt. Muss sie überhaupt etwas tun?
(c) Warum wäre es riskant, „monatliche Kosten" zu entfernen, ohne die kausale Struktur zu kennen?

Aufgabe 2.4 – Korrelation vs. VIF

Gegeben die Korrelationsmatrix:

	X_1	X_2	X_3
X_1	1,00	0,95	0,20
X_2	0,95	1,00	0,30
X_3	0,20	0,30	1,00

(a) Welches Variablenpaar ist offensichtlich problematisch?
(b) Approximieren Sie $R_1^2 \approx r_{12}^2 = 0{,}90$ und berechnen Sie VIF_1.
(c) Warum ist der wahre VIF vermutlich noch höher als Ihre Approximation?

Aufgabe 2.5 – Vorhersage vs. Interpretation

Erklären Sie in eigenen Worten:
(a) Warum schadet Multikollinearität der Interpretation einzelner Koeffizienten?
(b) Warum schadet sie der Vorhersagegenauigkeit weniger?
(c) Für welches der beiden Ziele (Vorhersage/Interpretation) ist es *wichtiger*, die Multikollinearität zu behandeln?

Kernaussage: Multikollinearität ist für die *Interpretation* einzelner Koeffizienten problematisch, weniger für die Vorhersage. Der VIF misst die Aufblähung der Varianz: $VIF > 5$ verdient Aufmerksamkeit, $VIF > 10$ erfordert Handlung. Die Wahl der Lösung (Entfernen, Kombinieren, Regularisieren, PCA, mehr Daten) hängt vom Analyseziel und der kausalen Struktur ab.

3 Das starre Navi – Bias-Varianz-Tradeoff

Lenas Herausforderung

Lena ist stolz: Ihr DataCo-Modell erreicht 99% Genauigkeit auf den Trainingsdaten. Jeder einzelne Kunde wird korrekt klassifiziert, jede Kündigung präzise vorhergesagt. Sie präsentiert die Ergebnisse ihrem Chef – und der fragt: "Wie gut funktioniert das bei neuen Kunden?" Lena testet auf dem zurückgehaltenen Testset: 52%. Kaum besser als eine Münze werfen. Der Chef ist nicht beeindruckt. Was ist passiert?

3.1 Entdeckungsfrage

Entdeckungsfrage

Routen speichern vs. Netz verstehen: Stellen Sie sich ein Navi vor, das jede einzelne Route perfekt gespeichert hat – jede Abbiegung, jede Einbahnstraße, jeder Kreisverkehr mit exakten Straßennamen. Trotzdem versagt es bei der nächsten Umleitung. Warum kennt das Navi nur alte Routen, obwohl es alles gespeichert hat?

3.2 Die Navi-Analogie

Overfitting: Ein Modell speichert die Trainingsdaten (inkl. Rauschen) statt das zugrundeliegende Muster zu erkennen.

Stellen Sie sich ein Navigationssystem vor, das jede Route Straße für Straße gespeichert hat: "Hauptstraße links, Bahnhofstraße geradeaus, dritte Ausfahrt im Kreisverkehr." Es kennt nicht die Logik des Straßennetzes, sondern nur die exakten Wege. Auf allen gespeicherten Strecken navigiert es perfekt – jede Abbiegung stimmt, jedes Ziel wird erreicht. Sein Trainings-Score: 100%.

Dann gibt es eine Baustelle. Die Hauptstraße ist gesperrt, das Navi muss eine Alternative finden. Aber es hat nie das Straßennetz verstanden, nur die Routen gespeichert. Es steht vor einem Rätsel – und schickt den Fahrer in eine Sackgasse. Sein Test-Score bricht ein.

Genau das passiert bei Lenas Modell. Ein zu komplexes Modell "merkt sich" die Trainingsdaten mitsamt ihrem Rauschen. Es kann perfekt reproduzieren, was es schon gesehen hat, aber es versagt, sobald auch nur leicht andere Daten kommen. Ein gutes Modell hingegen extrahiert das zugrundeliegende Muster – den systematischen Zusammenhang zwischen Features und Zielvariable – und ignoriert das zufällige Rauschen. Es "versteht das Straßennetz" statt "nur Routen zu speichern".

Der Unterschied zwischen Routen speichern und das Netz verstehen ist der Kern des Bias-Varianz-Tradeoffs. Ein Modell, das Routen speichert (Overfitting), hat zu viel Varianz; ein Modell, das zu stark vereinfacht (Underfitting), hat zu viel Bias.

3.3 Overfitting und Underfitting

Underfitting: Das Modell ist zu einfach, um den wahren Zusammenhang in den Daten abzubilden. Es "verpasst" systematisch wichtige Muster.

Overfitting und Underfitting sind die zwei Extremfehler der Modellierung. Um sie zu verstehen, hilft es, beide systematisch zu vergleichen.

Overfitting tritt auf, wenn ein Modell zu komplex ist – zu viele Parameter, zu viele Features oder zu viel Flexibilität. Das Modell passt sich den Trainingsdaten wie ein Maßanzug an: Jeder einzelne Datenpunkt wird perfekt getroffen, einschließlich zufälliger Schwankungen und Ausreißer. Die Folge ist ein sehr kleiner Trainingsfehler, aber ein viel größerer Testfehler. Die typischen Symptome sind:

- Die Trainingsgenauigkeit ist verdächtig hoch (oft $> 95\%$).
- Die Testgenauigkeit liegt deutlich darunter (oft 20–40 Prozentpunkte weniger).
- Die geschätzten Koeffizienten sind sehr groß (z. B. $\beta_1 = 42$, $\beta_2 = -87$).
- Das Modell hat viele Parameter relativ zur Anzahl der Beobachtungen.

Underfitting tritt auf, wenn ein Modell zu einfach ist, um den wahren Zusammenhang abzubilden. Ein lineares Modell für einen klar quadratischen Zusammenhang ist das klassische Beispiel. Das Modell verfehlt systematisch das Muster in den Daten. Die Symptome sind:

- Sowohl der Trainings- als auch der Testfehler sind hoch.
- Es gibt kein nennenswertes "Gap" zwischen Training und Test – aber beide sind schlecht.
- Das Modell ignoriert offensichtliche Strukturen in den Daten.

Das Ziel ist der Sweet Spot dazwischen: ein Modell, das komplex genug ist, um die echten Muster zu erfassen, aber einfach genug, um nicht das Rauschen mitzulernen.

3.4 In-Sample vs. Out-of-Sample

In-Sample-Fehler: Der Fehler auf den Trainingsdaten (auch: Trainingsfehler). Sinkt monoton mit steigender Modellkomplexität.

Out-of-Sample-Fehler: Der Fehler auf neuen, bisher nicht gesehenen Daten (auch: Testfehler, Generalisierungsfehler). Hat ein U-förmiges Minimum.

In der Praxis ist die zentrale Frage nicht "Wie gut passt mein Modell auf die Trainingsdaten?", sondern "Wie gut wird es bei neuen Daten funktionieren?" Der Unterschied zwischen diesen beiden Fragen ist der Unterschied zwischen In-Sample-Fehler und Out-of-Sample-Fehler.

Der In-Sample-Fehler (Trainingsfehler) gibt an, wie gut das Modell die Daten reproduziert, auf denen es trainiert wurde. Dieser Fehler sinkt monoton mit zunehmender Modellkomplexität – ein ausreichend komplexes Modell kann jede endliche Menge von Datenpunkten perfekt interpolieren. Bei Lenas DataCo-Modell lag der Trainingsfehler bei beeindruckenden 99%.

Der Out-of-Sample-Fehler (Testfehler) gibt an, wie gut das Modell bei bisher ungesehenen Daten abschneidet. Dieser Fehler hat typischerweise eine U-Form: Bei sehr einfachen Modellen ist er hoch (Underfitting), fällt dann bis zu einem Minimum (Sweet Spot) und steigt danach wieder an, wenn das Modell beginnt, Rauschen zu lernen (Overfitting). Bei Lenas Modell lag der Testfehler bei nur 52%.

Die **Lücke** zwischen Trainings- und Testfehler ist ein Diagnoseinstrument. Eine kleine Lücke (beide Fehler ähnlich hoch) deutet auf Underfitting hin. Eine große Lücke (kleiner Trainingsfehler, großer Testfehler) deutet auf Overfitting hin. Bei Lena: $99\% - 52\% = 47$ Prozentpunkte Lücke – ein dramatisches Overfitting.

3.5 Die Bias-Varianz-Zerlegung

Warum machen Modelle Fehler? Die Bias-Varianz-Zerlegung gibt eine elegante Antwort: Jeder Vorhersagefehler lässt sich in genau drei Quellen zerlegen.

Bias-Varianz-Zerlegung

Der erwartete quadratische Vorhersagefehler lässt sich zerlegen in:

$$E\left[(y - \hat{y})^2\right] = \underbrace{\text{Bias}^2}_{\text{Systematischer Fehler}} + \underbrace{\text{Varianz}}_{\text{Instabilität}} + \underbrace{\sigma^2}_{\text{Irreduzibler Fehler}}$$

Bias: Der systematische Fehler eines Modells. Entsteht, wenn das Modell zu einfach ist, um den wahren Zusammenhang abzubilden.

Varianz: Die Empfindlichkeit des Modells gegenüber dem konkreten Trainingsdatensatz. Ein instabiles Modell liefert bei verschiedenen Trainingssets sehr verschiedene Vorhersagen.

Irreduzibler Fehler: Das unvermeidbare Rauschen in den Daten (σ^2), das kein Modell eliminieren kann.

Betrachten wir die drei Terme im Detail. Der **Bias** (Verzerrung) misst, wie weit die durchschnittliche Vorhersage des Modells vom wahren Wert entfernt liegt, wenn man das Modell auf vielen verschiedenen Trainingssets trainieren würde. Ein hoher Bias bedeutet, dass das Modell den wahren Zusammenhang systematisch verfehlt – es ist zu einfach. Ein lineares Modell für einen quadratischen Zusammenhang hat hohen Bias, weil es die Krümmung niemals einfangen kann, egal wie viele Daten man ihm gibt. In der Formel wird der Bias quadriert, weil er eine gerichtete Abweichung ist: Die Quadrierung stellt sicher, dass positive und negative Abweichungen sich nicht gegenseitig aufheben, und große Abweichungen werden überproportional bestraft.

Die **Varianz** misst, wie stark die Vorhersagen des Modells schwanken, wenn man es auf verschiedenen Trainingssets trainiert. Ein Modell mit hoher Varianz reagiert empfindlich auf kleine Änderungen in den Trainingsdaten – es ist zu komplex. Ein Polynom vom Grad 15, trainiert auf 20 Datenpunkten, wird bei jedem neuen Trainingssample völlig andere Kurven produzieren. Es "merkt sich" die zufälligen Eigenheiten jedes einzelnen Datensatzes.

Der **irreduzible Fehler** σ^2 ist das Grundrauschen in den Daten. Selbst das perfekte Modell kann diesen Fehler nicht eliminieren, weil die Daten selbst unvorhersagbare Schwankungen enthalten. In Lenas Fall: Manche Kunden kündigen aus völlig unvorhersehbaren Gründen (Umzug, persönliche Krise), die in keinem Feature abgebildet sind.

Lenas DataCo-Zahlen

Angenommen, wir zerlegen den Fehler von Lenas Modell:

$$\text{Bias} = 0{,}4, \quad \text{Varianz} = 0{,}3, \quad \sigma = 0{,}2$$

Dann ergibt sich der Gesamtfehler als:

$$0{,}4^2 + 0{,}3 + 0{,}2^2 = 0{,}16 + 0{,}30 + 0{,}04 = 0{,}50$$

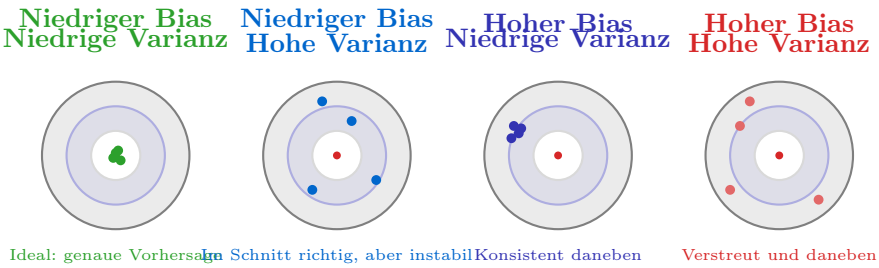
Bemerkenswertes Ergebnis: Die Hälfte des Fehlers kommt allein aus der Varianz! Der irreduzible Fehler trägt nur 0,04 bei. Lenas bester Hebel ist also, die Varianz zu reduzieren – durch ein einfacheres Modell, weniger Features oder Regularisierung.

Der **Tradeoff** entsteht, weil Bias und Varianz gegenläufig mit der Modellkomplexität zusammenhängen. Ein einfacheres Modell hat weniger Varianz (stabiler), aber mehr Bias (verfehlt den wahren Zusammenhang). Ein komplexeres Modell hat weniger Bias (flexibler), aber mehr Varianz (instabiler). Der optimale Punkt minimiert die Summe aus Bias^2 und Varianz – nicht einen der beiden Terme allein.

3.6 Die Zielscheiben-Analogie

Zielscheiben-Analogie: Veranschaulicht Bias (Abstand vom Zentrum) und Varianz (Streuung der Treffer) anhand einer Dartscheibe.

Die Zielscheiben-Analogie macht Bias und Varianz greifbar. Stellen Sie sich vor, Sie werfen mehrmals auf eine Dartscheibe. Das Zentrum der Scheibe ist der "wahre Wert", den wir vorhersagen wollen. Jeder Wurf ist eine Vorhersage auf einem anderen Trainingsdatensatz. Es gibt vier Kombinationen:

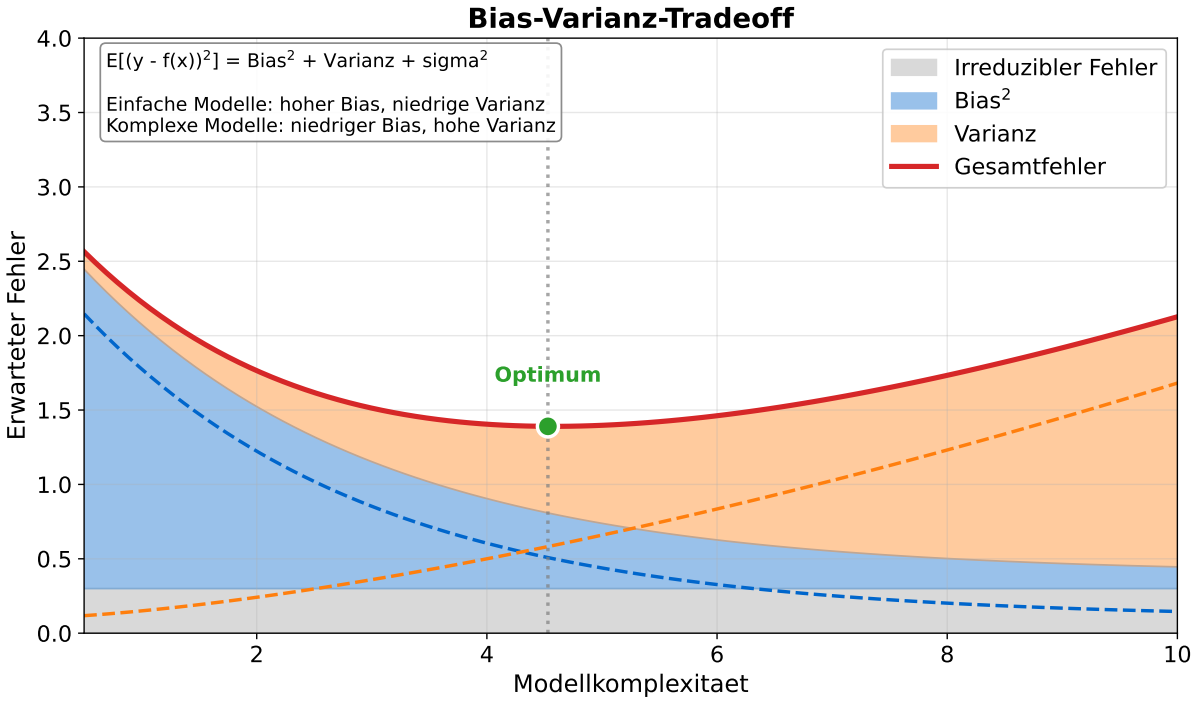


Niedriger Bias, niedrige Varianz (oben links) ist das Ideal: Die Treffer liegen eng beieinander und nahe am Zentrum. Das Modell trifft den wahren Zusammenhang und ist stabil.

Niedriger Bias, hohe Varianz (oben rechts) ist typisch für zu komplexe Modelle. Im Durchschnitt liegen die Vorhersagen richtig, aber bei jedem einzelnen Trainingset kommt etwas völlig anderes heraus. Das ist Lenas Polynom-Grad-15-Modell.

Hoher Bias, niedrige Varianz (Mitte rechts) ist typisch für zu einfache Modelle. Die Vorhersagen sind konsistent, aber systematisch daneben. Das wäre ein lineares Modell für Lenas quadratischen Zusammenhang.

Hoher Bias, hohe Varianz (rechts) ist der schlimmste Fall: Das Modell verfehlt den wahren Zusammenhang und ist dabei auch noch instabil.



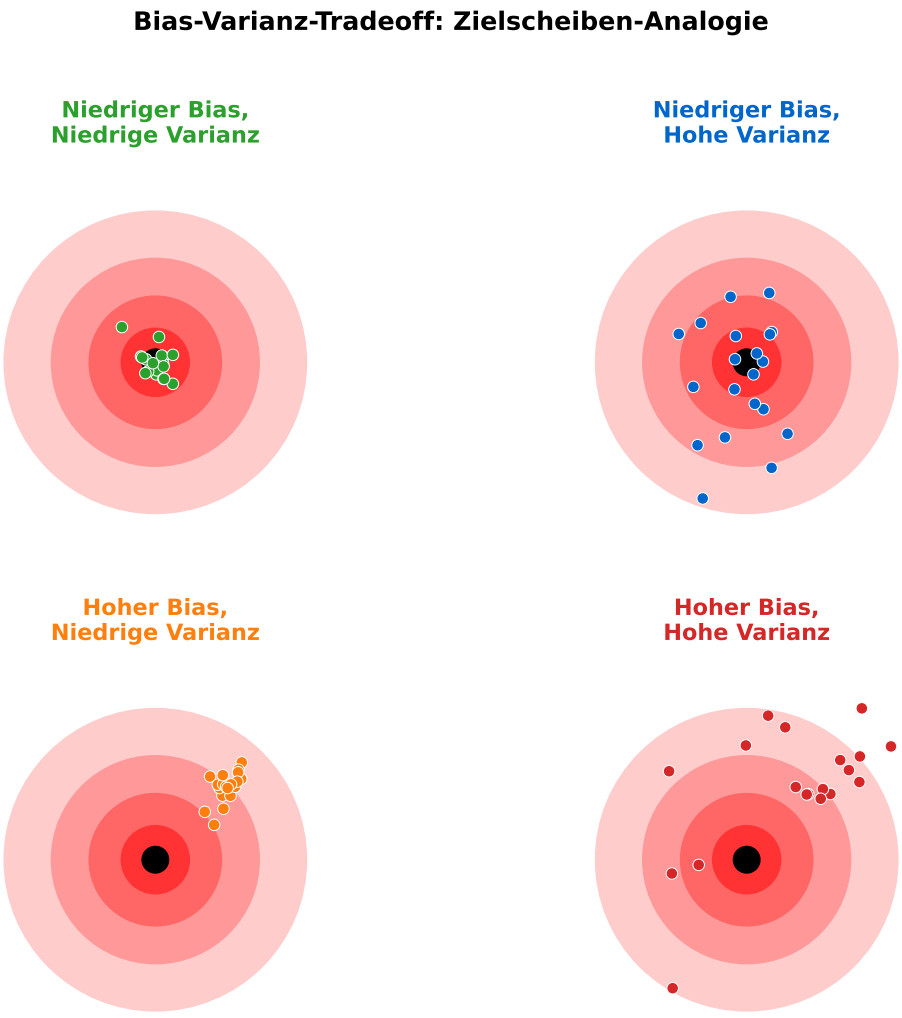


Abbildung 6: Bias-Varianz-Zielscheibe: Vier Szenarien im Überblick

3.7 Polynomgrade im Vergleich

Polynomiale Regression: Grad 1 vs. 3 vs. 15

Um den Bias-Varianz-Tradeoff konkret zu sehen, betrachten wir polynomiale Regression auf einem quadratischen Zusammenhang $y = 2 + 0{,}5x + 0{,}2x^2 + \varepsilon$.

Grad 1 (lineare Regression): Das Modell kann die Krümmung nicht abbilden. Der Trainingsfehler ist mäßig (MSE $\approx$ 5,1), der Testfehler ähnlich hoch (MSE $\approx$ 5,3). Diagnose: Underfitting, hoher Bias, niedrige Varianz.

Grad 3 (kubisches Polynom): Das Modell hat genug Flexibilität für die quadratische Struktur. Der Trainingsfehler sinkt (MSE $\approx$ 3,8), der Testfehler liegt nur leicht höher (MSE $\approx$ 4,1). Diagnose: Gute Balance, Sweet Spot.

Grad 15 (hochgradiges Polynom): Das Modell passt sich an jede kleine Schwankung an. Der Trainingsfehler ist winzig (MSE $\approx$ 2,5), aber der Testfehler explodiert (MSE $\approx$ 28,7). Die Koeffizienten werden riesig, die Kurve schlägt wild aus. Diagnose: Overfitting, niedrigerer Bias, aber enorme Varianz.

Die Lektion: Mehr Flexibilität heißt nicht automatisch bessere Vorhersagen. Der Testfehler hat ein U-förmiges Minimum.

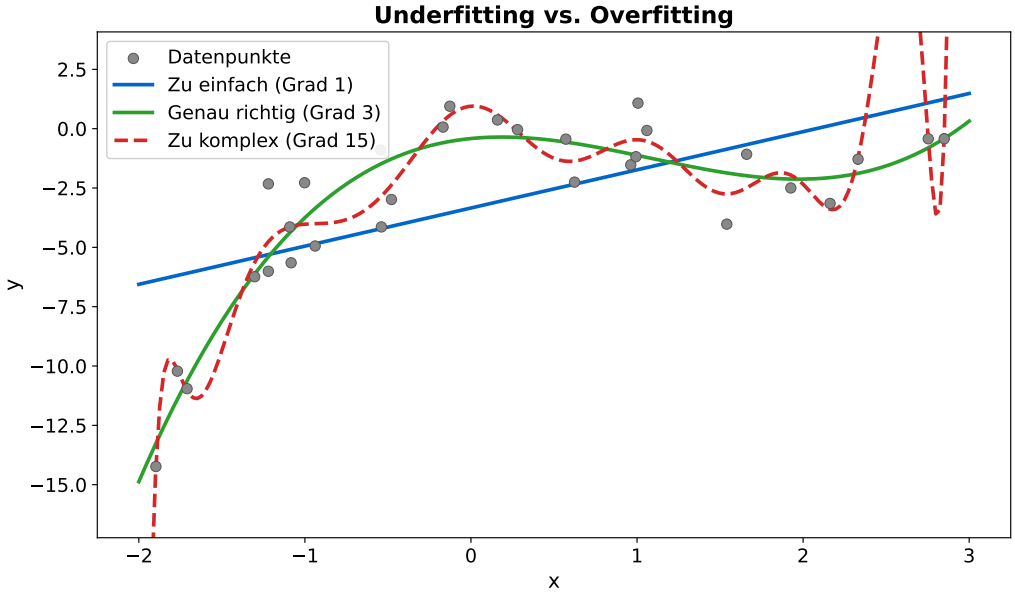


Abbildung 7: Polynomanpassung: Grad 1 (Underfitting), Grad 3 (gut), Grad 15 (Overfitting)

3.8 Faustregel: Beobachtungen pro Prädiktor

Faustregel: Mindestens 10–20 Beobachtungen pro Prädiktor im Modell, um Overfitting zu vermeiden.

Eine nützliche Faustregel aus der Praxis lautet: Für jede Variable im Modell sollten mindestens 10 bis 20 Beobachtungen vorhanden sein. Das Verhältnis n/p (Beobachtungen durch Prädiktoren) ist ein schneller Indikator für das Overfitting-Risiko.

Szenario	n	p	n/p **Ratio**
Lena: 4 Features	500	4	125 ✓ sicher
Lena: 50 Features	500	50	10 $\approx$ kritisch
Lena: 200 Features	500	200	2,5 $\times$ gefährlich
Lena: 1000 Features	500	1000	0,5 $\times\times$ unmöglich ohne Regularisierung

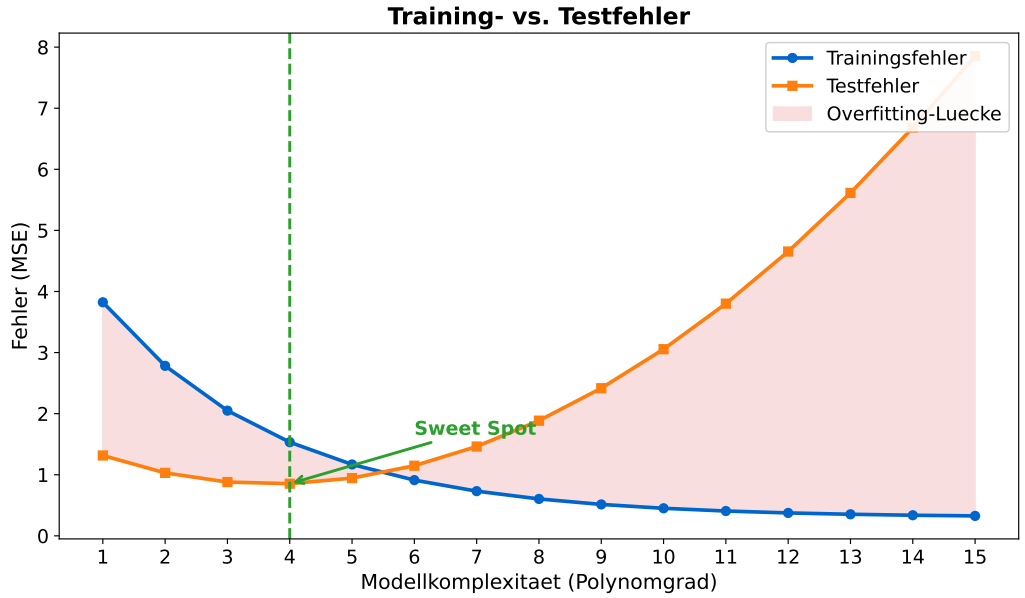


Abbildung 8: Training- vs. Testfehler als Funktion der Modellkomplexität

Was tun, wenn $p \gg n$ (mehr Features als Beobachtungen)? Die wichtigsten Strategien sind: Feature Selection (nur die relevantesten Variablen behalten), Regularisierung (Lasso, Ridge – das Thema von Kapitel 4), Dimensionsreduktion (z. B. PCA, das über den Rahmen dieser Vorlesung hinausgeht), oder schlicht mehr Daten sammeln, wenn das praktisch möglich ist.

Netflix Prize (2009) und Challenger (1986)

Der Netflix-Preis: Wenn Komplexität scheitert. Im Jahr 2006 lobte Netflix einen Preis von einer Million Dollar für das Team aus, das seinen Filmempfehlungs-Algorithmus um mindestens 10% verbessern konnte. Drei Jahre und Tausende von Teams später gewann 2009 das Team "BellKor's Pragmatic Chaos" – mit einem Ensemble aus über 100 verschiedenen Modellen, das 10,06% Verbesserung erreichte.
Die Pointe: Netflix setzte das Gewinnermodell nie in Produktion ein. Es war zu komplex, zu langsam und zu schwer zu warten. Ein einfacheres Modell mit nur drei Algorithmen war fast genauso gut – und 100 Mal schneller. Das ist der Bias-Varianz-Tradeoff in der Praxis: Das theoretisch beste Modell ist nicht immer das praktisch nützlichste.

Die Challenger-Katastrophe: Wenn Selection Bias tödlich wird. Am 28. Januar 1986 explodierte das Space Shuttle Challenger 73 Sekunden nach dem Start. Sieben Astronauten starben. Die Ursache: O-Ringe, die bei niedrigen Temperaturen spröde wurden.
Die NASA hatte Daten über O-Ring-Versagen bei früheren Flügen. Aber die Ingenieure machten einen fatalen Fehler: Sie analysierten nur die Flüge, bei denen es Schäden gab. Auf dieser gefilterten Datenbasis fanden sie keinen klaren Zusammenhang zwischen Temperatur und Versagen. Hätten sie alle Flüge einbezogen – auch die ohne Schäden –, wäre der Zusammenhang offensichtlich gewesen: Bei Temperaturen unter 18°C versagten die O-Ringe fast immer.
Lena bei DataCo: Wenn sie nur gekündigte Kunden analysiert und nicht auch die treuen Kunden, könnte sie das gleiche Muster verpassen. Selection Bias – die falsche Filterung von Daten – kann zu fatal falschen Schlussfolgerungen führen.

Häufige Fehler

Irrtum 1: "Mehr Features = besseres Modell." Nein. Jedes zusätzliche Feature erfordert mehr Daten, um zuverlässig geschätzt zu werden. Ab einem gewissen Punkt überwegt die erhöhte Varianz den reduzierten Bias. Dieses Phänomen heißt "Fluch der Dimensionalität" (Curse of Dimensionality): In hohen Dimensionen wird der Datenraum so dünn besetzt, dass keine Methode noch sinnvolle Muster erkennen kann.

Irrtum 2: "Training-Genauigkeit = Modellqualität." Nein. Die Training-Genauigkeit ist eine optimistische Überschätzung der tatsächlichen Leistung. Was zaehlt, ist ausschließlich die Out-of-Sample-Performance – also wie gut das Modell auf neuen, ungesehenen Daten funktioniert. Ein Modell mit 99% Training-Genauigkeit und 52% Test-Genauigkeit (wie bei Lena) ist schlechter als ein Modell mit 75% Training- und 73% Test-Genauigkeit.

Irrtum 3: "Overfitting tritt nur bei kleinen Datensätzen auf." Nein. Auch bei großen Datensätzen kann Overfitting auftreten, insbesondere wenn die Zahl der Features im Verhältnis zur Beobachtungszahl groß ist. Ein Datensatz mit 100.000 Beobachtungen und 50.000 Features ist genauso anfällig für Overfitting wie einer mit 100 Beobachtungen und 50 Features – das entscheidende Verhältnis n/p ist in beiden Fällen ähnlich.

3.9 R-Code: Overfitting demonstrieren

R-Code

```r
# Wahrer Zusammenhang: quadratisch
set.seed(123)
n_train <- 50; n_test <- 100

x_train <- runif(n_train, 0, 10)
y_train <- 2 + 0.5*x_train + 0.2*x_train^2 + rnorm(n_train, 0, 2)

x_test <- runif(n_test, 0, 10)
y_test <- 2 + 0.5*x_test + 0.2*x_test^2 + rnorm(n_test, 0, 2)

train_data <- data.frame(x = x_train, y = y_train)
test_data  <- data.frame(x = x_test,  y = y_test)

# Verschiedene Polynomgrade testen
degrees <- 1:15
train_mse <- test_mse <- numeric(15)

for (d in degrees) {
  fit <- lm(y ~ poly(x, d, raw = TRUE), data = train_data)
  train_mse[d] <- mean((train_data$y - predict(fit, train_data))
     ^2)
  test_mse[d]  <- mean((test_data$y  - predict(fit, test_data))
     ^2)
}

# Ergebnis: Trainings-MSE sinkt immer, Test-MSE hat U-Form
results <- data.frame(Grad = degrees,
                      Train_MSE = round(train_mse, 2),
                      Test_MSE  = round(test_mse, 2))
print(results)

# Optimaler Grad
cat("Optimaler Grad:", which.min(test_mse), "\n")
```

Interpretation: Der Trainingsfehler sinkt monoton mit steigendem Polynomgrad – mehr Flexibilität heißt bessere Anpassung an die Trainingsdaten. Der Testfehler hingegen erreicht sein Minimum bei Grad 2 oder 3 (dem wahren Zusammenhang) und steigt danach steil an, weil das Modell Rauschen mitlernt.

3.10 Übungsaufgaben

Aufgabe 3.1: Symptome erkennen

Lenas Kollegin zeigt ein Modell mit Trainings-$R^2 = 0{,}98$ und Test-$R^2 = 0{,}41$. Was ist das Problem, und welche drei konkreten Maßnahmen würden Sie empfehlen?

Aufgabe 3.2: Bias-Varianz-Zerlegung

Für ein Modell gelten: Bias = 0,6, Varianz = 0,1, $\sigma = 0{,}3$.

a) Berechnen Sie den Gesamtfehler.

b) Welche Komponente dominiert?

c) Sollte Lena das Modell einfacher oder komplexer machen?

Aufgabe 3.3: Zielscheibe zuordnen

Ordnen Sie die folgenden Modelle den vier Zielscheiben-Quadranten zu:

a) Lineare Regression auf einen quadratischen Zusammenhang (viele Daten)

b) Polynom Grad 20 auf 15 Datenpunkte

c) Polynom Grad 3 auf einen quadratischen Zusammenhang (viele Daten)

d) Konstantes Modell $\hat{y} = \bar{y}$ mit nur 5 Datenpunkten

Aufgabe 3.4: Faustregel anwenden

Lena hat $n = 300$ Beobachtungen. Wie viele Prädiktoren kann sie maximal ins Modell nehmen, ohne die Faustregel zu verletzen? Was passiert, wenn ihr Chef 80 Features verlangt?

Aufgabe 3.5: Netflix vs. Challenger

Vergleichen Sie den Netflix-Prize-Fall und die Challenger-Katastrophe.

a) In welchem Fall spielte Overfitting die Hauptrolle, in welchem Selection Bias?

b) Was hätte jeweils geholfen? Nennen Sie je eine konkrete Maßnahme.

Kernaussage: Das Ziel ist minimaler Gesamtfehler, nicht minimaler Bias oder minimale Varianz. Optimale Modellkomplexität balanciert beides. In der Praxis: Immer auf einem separaten Testset evaluieren und die Lücke zwischen Trainings- und Testfehler beobachten.

4 Die Leine für wilde Koeffizienten – Regularisierung

Lenas Herausforderung

Lena schaut sich die Koeffizienten ihres DataCo-Modells an: $\beta_1 = 42$, $\beta_2 = -87$, $\beta_3 = 63$. Die Zahlen sind riesig und wechseln bei jedem neuen Trainingsset wild ihre Vorzeichen. Das Modell hat das Rauschen in den Daten gespeichert statt das Muster zu erkennen, und die explodierten Koeffizienten sind das Symptom. Lena braucht eine Leine für ihre wilden Koeffizienten.

4.1 Entdeckungsfrage

Entdeckungsfrage

Wie zaehmt man Koeffizienten, die aus dem Ruder laufen? Was wäre, wenn man ein "Budget" für die Koeffizientengröße hätte – wenn jeder zusätzliche Betrag an Koeffizientengröße Kosten verursacht?

4.2 Die Leine und der Türsteher

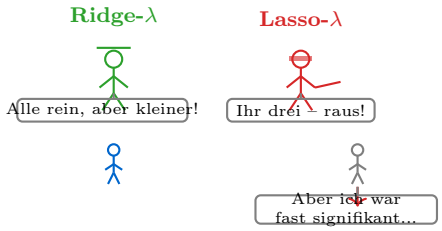
Regularisierung: Eine Technik, die große Koeffizienten bestraft, um Overfitting zu reduzieren und die Generalisierung zu verbessern.

Zwei Analogien helfen, die Regularisierung zu verstehen:

Die Leine. Stellen Sie sich Lenas Koeffizienten als Hunde vor, die an der Leine geführt werden. Ohne Leine rennen sie in alle Richtungen – β_1 springt auf 42, β_2 rast auf -87. Das ist ein unkontrolliertes Modell ohne Regularisierung: Es passt sich an jedes kleine Rauschen an. Die Regularisierung ist die Leine. Sie lässt den Hunden etwas Bewegungsfreiheit (die Koeffizienten dürfen nicht null sein), zieht sie aber zurück, wenn sie zu weit ausscheren. Je kürzere Leine (größeres λ), desto weniger Freiheit für die Koeffizienten.

Der Türsteher. Stellen Sie sich Lambda (λ) als Türsteher vor einem Club vor. Die Koeffizienten stehen in der Schlange und wollen rein. Hier unterscheiden sich Ridge und Lasso:

- **Ridge** (λ als hoeflicher Türsteher): Lässt alle Koeffizienten rein, aber schrumpft jeden ein wenig. Wer vorher groß war, ist immer noch größer als die anderen – aber alle sind kleiner geworden. "Alle rein, aber bitte etwas bescheidener!"
- **Lasso** (λ als strenger Türsteher mit Sonnenbrille): Prüft jeden Koeffizienten an der Tuer. Wer zu unwichtig ist, wird abgewiesen – komplett auf Null gesetzt. "Du – raus! Du – raus! Ihr drei duerft rein." Ein abgewiesener Koeffizient murmelt: "Aber ich war fast signifikant..."



4.3 Die Strafidee

Strafterm (Penalty): Ein zusätzlicher Term in der Verlustfunktion, der große Koeffizienten "teuer" macht und damit Einfachheit belohnt.

Die Kernidee der Regularisierung lässt sich in einem Satz zusammenfassen: Fuege zur Verlustfunktion einen Strafterm hinzu, der große Koeffizienten bestraft. Das Modell muss jetzt zwei Ziele gleichzeitig optimieren – gute Datenanpassung *und* kleine Koeffizienten.

Regularisierte Verlustfunktion

$$\min_{\beta} \underbrace{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}_{\text{Anpassung (RSS)}} + \underbrace{\lambda \cdot \text{Strafe}(\beta)}_{\text{Komplexitätskosten}}$$

Lambda (λ): Der Regularisierungsparameter. Steuert die Stärke der Bestrafung. $\lambda = 0$: keine Bestrafung (OLS). $\lambda \to \infty$: alle Koeffizienten gegen Null.

Der erste Term ist die vertraute Residuenquadratsumme (RSS) – er misst, wie gut das Modell die Daten anpasst. Der zweite Term ist neu: Er misst die "Größe" der Koeffizienten und bestraft große Werte. Der Parameter λ (Lambda) steuert die Balance zwischen beiden Zielen.

Hier ist das Spektrum von λ entscheidend. Am einen Ende steht $\lambda = 0$: Keine Strafe, das Modell reduziert sich auf die gewöhnliche OLS-Regression. Alle Koeffizienten dürfen so groß werden, wie sie wollen. Am anderen Ende steht $\lambda \to \infty$: Maximale Strafe, alle Koeffizienten werden gegen Null gedrückt, weil jeder Betrag an Koeffizientengröße "unendlich teuer" wird. Dazwischen liegt der optimale λ-Wert, der die beste Balance zwischen Datenanpassung und Einfachheit findet.

Die Analogie zum Budget ist hilfreich: Stellen Sie sich vor, jeder Koeffizient hat einen "Preis". Ohne Budget-Beschränkung ($\lambda = 0$) kaufen Sie alles, was die Anpassung verbessert – auch Features, die nur Rauschen einfangen. Mit straffem Budget (großes λ) müssen Sie priorisieren und verzichten auf die weniger nützlichen "Ausgaben".

4.4 Ridge Regression (L2)

Ridge Regression: Regularisierung mit L2-Strafe ($\sum \beta_j^2$). Schrumpft alle Koeffizienten, setzt aber keinen auf exakt Null.

Ridge Regression verwendet die quadratische Strafe (L2-Norm):

Ridge Regression (L2-Penalty)

$$\min_{\beta} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$

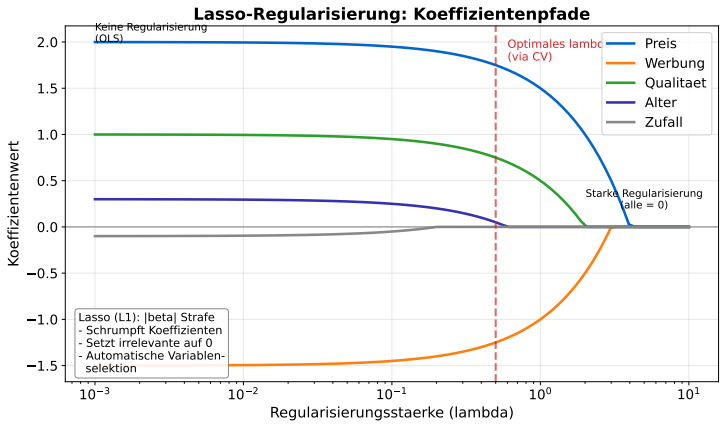
Dabei ist β_j der Koeffizient der j-ten Variable und p die Gesamtzahl der Prädiktoren. Der Strafterm $\lambda \sum \beta_j^2$ bestraft das Quadrat jedes Koeffizienten.

Die Eigenschaften von Ridge sind:

Schrumpfung ohne Selektion. Ridge schrumpft alle Koeffizienten Richtung Null, aber keiner wird exakt Null. Selbst ein Feature, das nur sehr wenig zum Modell beiträgt, behält einen kleinen, von Null verschiedenen Koeffizienten. Das bedeutet: Alle Features bleiben im Modell.

Stabilisierung bei Multikollinearität. Wenn Features stark korreliert sind (wie Lenas "Vertragslaufzeit" und "monatliche Kosten" aus Kapitel 2), werden die OLS-Koeffizienten instabil. Ridge stabilisiert sie, indem es die Freiheit der Koeffizienten einschränkt.

Geometrische Intuition. Die L2-Strafe definiert eine kreisförmige Nebenbedingung im Koeffizientenraum. Die OLS-Lösung (dargestellt als Ellipsen gleichen Fehlers) wird auf diesen Kreis projiziert. Da ein Kreis keine Ecken hat, trifft die Projektion fast nie eine Achse – deshalb werden Koeffizienten nicht exakt Null.



4.5 Lasso Regression (L1)

Lasso (Least Absolute Shrinkage and Selection Operator): Regularisierung mit L1-Strafe ($\sum |\beta_j|$). Schrumpft Koeffizienten und setzt unwichtige auf exakt Null.

Lasso (Least Absolute Shrinkage and Selection Operator) verwendet die Absolutwert-Strafe (L1-Norm):

Lasso Regression (L1-Penalty)

$$\min_{\beta} \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

Der Strafterm $\lambda \sum |\beta_j|$ bestraft den Absolutwert jedes Koeffizienten.

Der entscheidende Unterschied zu Ridge: Lasso setzt unwichtige Koeffizienten auf **exakt Null**. Es führt also automatisch eine Variablenselektion durch – es wähltdie wichtigsten Features aus und entfernt den Rest.

Warum erzwingt L1 Nullen? Die geometrische Erklärung ist elegant. Die L1-Nebenbedingung ist nicht mehr ein Kreis, sondern ein Diamant (Raute). Ein Diamant hat Ecken, und an den Ecken ist mindestens eine Koordinate gleich Null. Die OLS-Lösung (Ellipsen) trifft den Diamant fast immer an einer Ecke – und genau dort wird mindestens ein Koeffizient zu Null.

Die Analogie zur Budget-Entscheidung macht es griffig: Ridge sagt "Gib überall etwas weniger aus" (der Kreis ist glatt, kein Projekt wird komplett gestrichen). Lasso sagt "Streich ganze Projekte" (der Diamant hat Ecken, an den Ecken wird mindestens ein Posten komplett eliminiert).

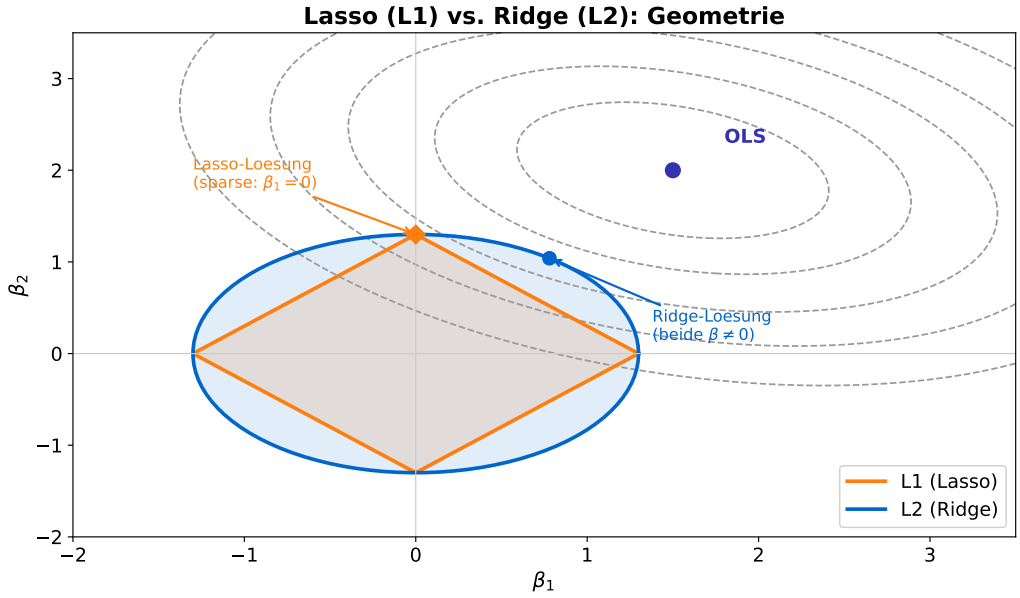


Abbildung 9: Geometrie der Regularisierung: L2-Kreis (Ridge) vs. L1-Raute (Lasso)

Die Eigenschaften von Lasso im Überblick:

- Schrumpft Koeffizienten *und* setzt unwichtige auf exakt Null.
- Erzeugt automatisch sparsame ("sparse") Modelle – ideal, wenn nur wenige Features wirklich relevant sind.
- Verbessert die Interpretierbarkeit: Weniger Koeffizienten = einfacheres Modell.

- Nachteil: Bei stark korrelierten Features wähltLasso oft willkürlich eines aus und setzt die anderen auf Null.

4.6 Elastic Net

Elastic Net: Kombination aus L1- und L2-Strafe. Verbindet die Selektion von Lasso mit der Stabilität von Ridge.

Elastic Net ist die Kombination aus Ridge und Lasso, vorgeschlagen von Zou und Hastie (2005). Es verwendet beide Strafterme gleichzeitig:

Elastic Net

In der `glmnet`-Parametrisierung:

$$\min_{\beta} \sum (y_i - \hat{y}_i)^2 + \lambda \Big[\alpha \sum_{j=1}^{p} |\beta_j| + (1-\alpha) \sum_{j=1}^{p} \beta_j^2 \Big]$$

Dabei steuert $\alpha \in [0,1]$ die Mischung: $\alpha = 1$ ist reines Lasso, $\alpha = 0$ ist reines Ridge, und $\alpha = 0{,}5$ ist eine 50/50-Mischung beider Strafen.

Hinweis: In manchen Lehrbüchern und auf den Vorlesungsfolien wird die äquivalente Zwei-Lambda-Form $\lambda_1 \sum |\beta_j| + \lambda_2 \sum \beta_j^2$ verwendet. Die `glmnet`-Form mit einem λ und einem Mischungsparameter α ist in der Praxis gängiger, da `cv.glmnet` den optimalen λ-Wert per Kreuzvalidierung bestimmt.

Wann Elastic Net? In der Praxis ist Elastic Net oft die sicherste Wahl, wenn man sich nicht sicher ist, ob Ridge oder Lasso besser passt. Es kombiniert die Variablenselektion von Lasso mit der Stabilität von Ridge bei korrelierten Features.

4.7 Vergleichstabelle: Ridge vs. Lasso vs. Elastic Net

Eigenschaft	**Ridge**	**Lasso**	**Elastic Net**
Strafterm	$\lambda \sum \beta_j^2$	$\lambda \sum \lvert\beta_j\rvert$	$\lambda[\alpha \sum \lvert\beta_j\rvert + (1-\alpha) \sum \beta_j^2]$
Koeffizienten auf 0?	Nein	Ja	Ja
Variablenselektion?	Nein	Ja	Ja
Multikollinearität?	Sehr gut	Problematisch	Gut
Viele kleine Effekte?	Gut	Weniger gut	Gut
Wenige große Effekte?	Weniger gut	Sehr gut	Gut
`glmnet` α	$\alpha = 0$	$\alpha = 1$	$\alpha \in (0,1)$

Wann Ridge? Wenn Sie erwarten, dass viele Features kleine, aber echte Effekte haben und alle im Modell bleiben sollen. Typisch bei Multikollinearität (korrelierte Features), wo Lasso willkürlich Features auswählt.

Wann Lasso? Wenn Sie erwarten, dass nur wenige Features wirklich relevant sind und der Rest Rauschen ist. Lasso erzeugt automatisch ein sparsames Modell.

Wann Elastic Net? Wenn Sie unsicher sind oder wenn korrelierte Feature-Gruppen vorliegen. Elastic Net wähltGruppen korrelierter Features gemeinsam aus, statt willkürlich eine auszusuchen.

4.8 Lambda wählen

Kreuzvalidierung (CV): Methode zur Wahl des optimalen λ: Daten werden in k Folds aufgeteilt, jedes Fold dient einmal als Testset.

`lambda.min`: Der λ-Wert, der den kleinsten CV-Fehler erzielt. Optimiert rein auf Vorhersagegenauigkeit.

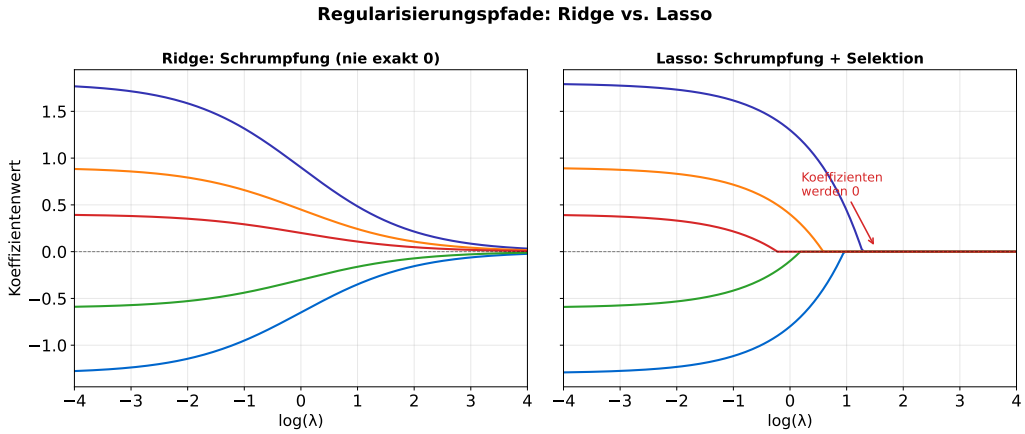


Abbildung 10: Koeffizientenpfade im Vergleich: Ridge (stetige Schrumpfung) vs. Lasso (Selektion)

`lambda.1se`: Der größte λ-Wert, dessen CV-Fehler noch innerhalb eines Standardfehlers von `lambda.min` liegt. Erzeugt ein sparsameres Modell.

Die zentrale Frage bei der Regularisierung lautet: Welchen Wert soll λ haben? Die Antwort: Kreuzvalidierung. Die Funktion `cv.glmnet` in R führt automatisch k-Fold-CV durch und liefert zwei wichtige λ-Werte:

`lambda.min` ist der λ-Wert, der den kleinsten mittleren CV-Fehler (MSE) erzielt. Er optimiert rein auf Vorhersagegenauigkeit.

`lambda.1se` ist der größte λ-Wert, dessen CV-Fehler noch innerhalb eines Standardfehlers des Minimums liegt. Dieser Wert führt zu einem einfacheren Modell (mehr Schrumpfung, bei Lasso mehr Nullen) mit nur minimal schlechterer Vorhersage. In der Praxis wird oft `lambda.1se` bevorzugt, weil es dem Prinzip der Sparsamkeit (Occam's Razor) folgt.

Die **Koeffizientenpfade** zeigen grafisch, wie sich die Koeffizienten verändern, wenn λ zunimmt. Bei Ridge schrumpfen alle Koeffizienten sanft gegen Null, aber keiner verschwindet. Bei Lasso fallen die Koeffizienten nacheinander auf exakt Null – die weniger wichtigen zuerst. Diese Grafiken sind ein wertvolles Diagnosewerkzeug, um zu verstehen, welche Features bei welcher Stärke der Regularisierung überleben.

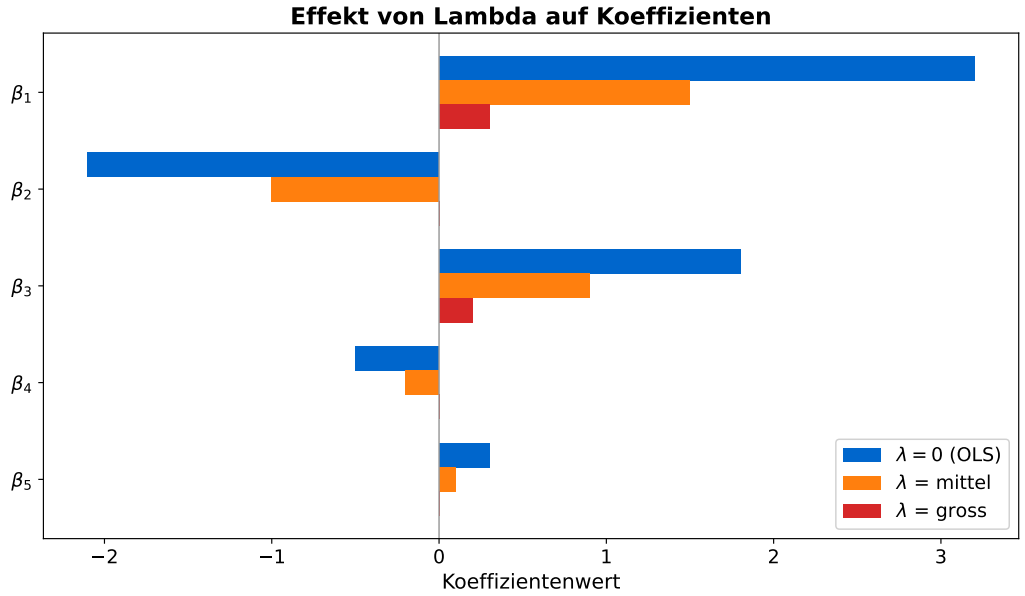


Abbildung 11: Effekt von λ: Koeffizienten schrumpfen mit steigendem Strafterm

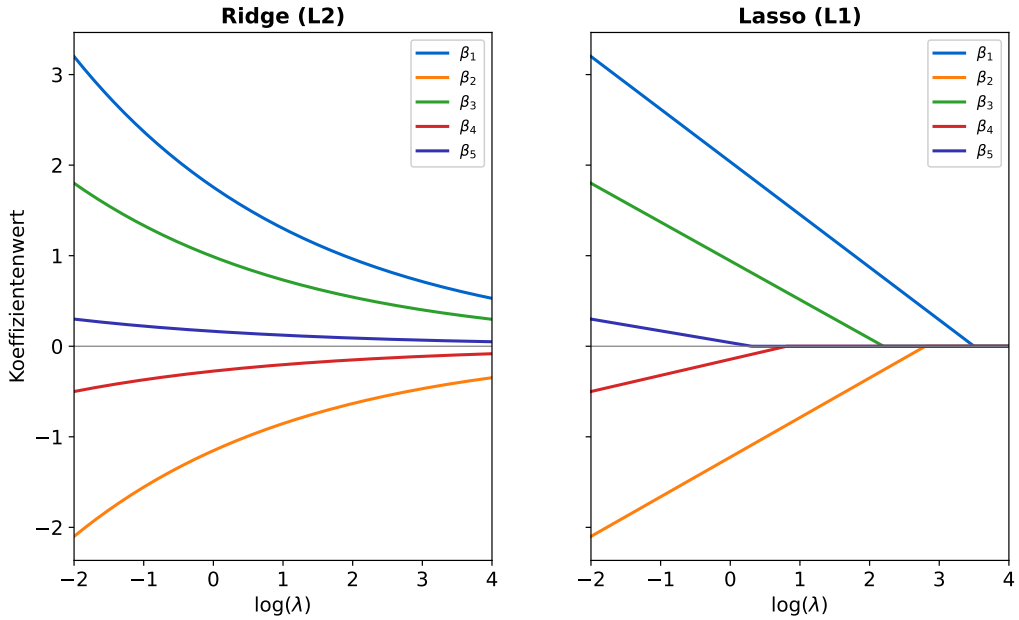


Abbildung 12: Koeffizientenpfade: Ridge (links) vs. Lasso (rechts) bei steigendem λ

4.9 Beispiele mit Zahlen

Lenas glmnet-Pipeline bei DataCo

Lena hat 500 Kunden und 12 Features (Alter, Vertragslaufzeit, monatliche Kosten, Anzahl Beschwerden, Premium-Status, Nutzungshäufigkeit, und weitere). Ohne Regularisierung explodieren einige Koeffizienten. Sie setzt Lasso mit Kreuzvalidierung ein:

1. **Vorbereitung**: Modellmatrix erstellen mit `model.matrix()`, Zielvariable extrahieren.
2. **CV**: `cv.glmnet(x, y, alpha = 1, nfolds = 10)` liefert `lambda.min` = 0,023.
3. **Koeffizienten**: Bei `lambda.min` werden 3 der 12 Koeffizienten auf exakt Null gesetzt: "PLZ" (Postleitzahl), "Browser-Typ" und "Anmelde-Wochentag". Diese Features tragen nicht zur Kündigungsvorhersage bei.
4. **Ergebnis**: Die verbleibenden 9 Features ergeben ein interpretierbares Modell. Die wichtigsten Prädiktoren sind "Anzahl Beschwerden" ($\beta = 0{,}31$), "Nutzungshäufigkeit" ($\beta = -0{,}28$) und "Vertragslaufzeit" ($\beta = -0{,}19$).

Ohne Lasso hätte Lenas Modell 12 Koeffizienten mit teilweise absurden Werten ($\beta_{\text{PLZ}} = 3{,}2$, $\beta_{\text{Browser}} = -7{,}8$). Mit Lasso: schlankes, interpretierbares Modell mit besserer Out-of-Sample-Performance.

Vorher/Nachher: 12 Features, Lasso setzt 3 auf Null

Feature	OLS $\hat{\beta}$	Lasso $\hat{\beta}$	Status
Alter	0.02	0.01	geschrumpft
Vertragslaufzeit	−0.25	−0.19	geschrumpft
Monatl. Kosten	0.18	0.12	geschrumpft
Beschwerden	0.35	0.31	geschrumpft
Premium-Status	−0.42	−0.24	geschrumpft
Nutzungshäufigkeit	−0.33	−0.28	geschrumpft
Support-Kontakte	0.21	0.15	geschrumpft
Online-Aktivität	−0.15	−0.08	geschrumpft
Vertragswert	0.11	0.04	geschrumpft
PLZ	3.20	0.00	entfernt
Browser-Typ	−7.80	0.00	entfernt
Anmelde-Wochentag	1.50	0.00	entfernt

Die Koeffizienten bei OLS sind teilweise absurd groß (Browser-Typ: −7,8!) und instabil. Lasso identifiziert drei irrelevante Features und schrumpft die restlichen auf vernuenftige Größenordnungen.

Von Hoerl & Kennard (1970) über Tibshirani (1996) zu Zou & Hastie (2005)

Die Geschichte der Regularisierung erstreckt sich über 35 Jahre und drei Durchbrüche:
1970: Ridge Regression von Arthur Hoerl und Robert Kennard. Hoerl arbeitete als Chemiker bei DuPont und hatte ein praktisches Problem: Seine Messdaten enthielten viele korrelierte Variablen (verschiedene chemische Messungen am selben Prozess), und die OLS-Regression lieferte instabile, absurde Koeffizienten. Zusammen mit dem Statistiker Kennard entwickelte er die Idee, die Koeffizienten durch eine L2-Strafe zu schrumpfen. Der Name "Ridge" kommt vom "Grat" (Ridge), der entsteht, wenn man zur Diagonale der Matrix $X^T X$ einen kleinen positiven Wert addiert.
1996: Lasso von Robert Tibshirani (Stanford). Tibshiranis Durchbruch bestand darin, die L1-Strafe zu verwenden statt der L2-Strafe. Die mathematische Konsequenz war überraschend: Durch die Ecken des L1-Diamanten werden manche Koeffizienten exakt auf Null gesetzt. Das Modell führt also automatisch eine Variablenselektion durch – etwas, das Ridge nicht kann. Der Name "Lasso" steht für "Least Absolute Shrinkage and Selection Operator".
2005: Elastic Net von Hui Zou und Trevor Hastie (ebenfalls Stanford). Sie erkannten, dass Lasso bei korrelierten Features problematisch ist (es wähltwillkürlich eines aus) und kombinierten beide Strafterme. Elastic Net vereint die Variablenselektion von Lasso mit der Stabilität von Ridge.
Alle drei Methoden sind heute in der R-Bibliothek `glmnet` implementiert – eine der meistgenutzten Bibliotheken in der statistischen Praxis.

Häufige Fehler

Irrtum 1: "Regularisierung verbessert immer das Modell." Nein. Regularisierung hilft nur bei Overfitting. Wenn das Modell bereits underfittet (zu einfach), würde zusätzliche Regularisierung die Koeffizienten weiter schrumpfen und das Modell noch schlechter machen. Regularisierung ist ein Werkzeug gegen zu viel Varianz – nicht gegen zu viel Bias.

Irrtum 2: "Lasso ist immer besser als Ridge." Nein. Lasso ist besser, wenn tatsächlich nur wenige Features relevant sind (Sparsity). Aber wenn viele Features kleine, aber echte Effekte haben, schneidet Ridge besser ab, weil es alle Features behält und ihre Beiträge zusammen nutzt. Bei stark korrelierten Features kann Lasso willkürlich eines auswählen und die anderen ignorieren, während Ridge alle gleichmäßig schrumpft.

Irrtum 3: "Lambda sollte so groß wie möglich sein." Nein. Ein zu großes λ toetet alles Signal und drückt alle Koeffizienten gegen Null. Das Ergebnis ist ein Modell, das kaum besser als der Mittelwert vorhersagt. Das optimale λ balanciert Anpassung und Einfachheit – deshalb wird es per Kreuzvalidierung gewählt, nicht willkürlich festgelegt.

4.10 R-Code: glmnet Pipeline

R-Code

```r
library(glmnet)

# --- Daten vorbereiten ---
# 200 Beobachtungen, 20 Features, nur 5 relevant
set.seed(42)
n <- 200; p <- 20
X <- matrix(rnorm(n * p), n, p)
colnames(X) <- paste0("x", 1:p)

# Wahre Koeffizienten: nur x1-x5 haben Effekt
true_beta <- c(3, -2, 1.5, -1, 0.5, rep(0, 15))
y <- X %*% true_beta + rnorm(n, 0, 2)

# --- Lasso mit Kreuzvalidierung ---
cv_lasso <- cv.glmnet(X, y, alpha = 1, nfolds = 10)

# Zwei wichtige Lambda-Werte
cat("lambda.min:", round(cv_lasso$lambda.min, 4), "\n")
cat("lambda.1se:", round(cv_lasso$lambda.1se, 4), "\n")

# Koeffizienten bei lambda.min
coef_lasso <- coef(cv_lasso, s = "lambda.min")
print(coef_lasso)

# Wie viele wurden auf Null gesetzt?
cat("Null-Koeffizienten:", sum(coef_lasso[-1] == 0),
    "von", p, "\n")

# --- Ridge zum Vergleich ---
cv_ridge <- cv.glmnet(X, y, alpha = 0, nfolds = 10)
coef_ridge <- coef(cv_ridge, s = "lambda.min")

cat("Ridge Null-Koeffizienten:",
    sum(coef_ridge[-1] == 0), "von", p, "\n")

# --- Koeffizientenpfade plotten ---
lasso_fit <- glmnet(X, y, alpha = 1)
plot(lasso_fit, xvar = "lambda", label = TRUE)
title("Lasso: Koeffizientenpfade")

# --- Elastic Net ---
cv_enet <- cv.glmnet(X, y, alpha = 0.5, nfolds = 10)
coef_enet <- coef(cv_enet, s = "lambda.min")
cat("Elastic Net Null-Koeffizienten:",
    sum(coef_enet[-1] == 0), "von", p, "\n")
```

Interpretation: Lasso erkennt in der Regel korrekt, dass nur die ersten 5 Features echte Effekte haben, und setzt die meisten der 15 Rausch-Features auf Null. Ridge behält alle 20 Features mit kleinen, von Null verschiedenen Koeffizienten. Elastic Net liegt dazwischen – es setzt einige auf Null, aber weniger aggressiv als reines Lasso.

4.11 Übungsaufgaben

Aufgabe 4.1: Strafterm verstehen

Erklären Sie in eigenen Worten, warum die L1-Strafe (Lasso) Koeffizienten auf exakt Null setzt, die L2-Strafe (Ridge) aber nicht. Nutzen Sie die Diamant-vs.-Kreis-Geometrie.

Aufgabe 4.2: Lambda-Spektrum

Beschreiben Sie, was mit den Koeffizienten eines Lasso-Modells passiert, wenn λ schrittweise von 0 auf einen sehr großen Wert erhöht wird. Welches Feature "überlebt" am laengsten? Warum?

Aufgabe 4.3: Ridge oder Lasso?

Für jede der folgenden Situationen: Würden Sie Ridge, Lasso oder Elastic Net empfehlen? Begründen Sie.

a) 500 Kunden, 8 Features, alle aus der Fachliteratur begründet.

b) 500 Kunden, 200 Features, die meisten vermutlich irrelevant.

c) 500 Kunden, 15 Features, davon 5 Gruppen mit jeweils 3 stark korrelierten Features.

Aufgabe 4.4: lambda.min vs. lambda.1se

Erklären Sie den Unterschied zwischen `lambda.min` und `lambda.1se`. In welcher Situation würde Lena `lambda.1se` bevorzugen? Wann wäre `lambda.min` die bessere Wahl?

Aufgabe 4.5: Lenas glmnet-Ergebnis interpretieren

Lenas `cv.glmnet`-Lauf für ihr Churn-Modell ergibt:

- `lambda.min = 0.023`, `lambda.1se = 0.089`
- Bei `lambda.min`: 9 von 12 Koeffizienten $\neq 0$
- Bei `lambda.1se`: 5 von 12 Koeffizienten $\neq 0$

a) Was bedeuten die zwei verschiedenen Lambda-Werte inhaltlich?

b) Welches Lambda würde Lena ihrem Chef präsentieren, wenn dieser ein möglichst einfaches Modell will?

c) Was ist der Preis für das einfachere Modell?

Kernaussage: Regularisierung opfert bewusst etwas Bias (die Koeffizienten werden Richtung Null "verzerrt"), gewinnt dafür aber deutlich an Stabilität (weniger Varianz). Ridge schrumpft alle, Lasso selektiert, Elastic Net kombiniert beides. Der Regularisierungsparameter λ wird immer per Kreuzvalidierung gewählt – niemals per Hand.

5 Rotierende Richter – Kreuzvalidierung

Lenas Herausforderung

Lena hat ihr DataCo-Churn-Modell auf einem Testset geprüft und bekommt 78 % Accuracy. Ihr Chef gratuliert, aber Lena ist skeptisch: Als sie den Datensatz neu mischt und einen anderen 80/20-Split wählt, sinkt die Accuracy auf 71 %. Beim dritten Versuch steigt sie auf 82 %. Drei Splits, drei verschiedene Zahlen – welche Genauigkeit stimmt jetzt?

5.1 Entdeckungsfrage

Entdeckungsfrage

Warum reicht ein einziger Train/Test-Split nicht aus? Stellen Sie sich vor, Sie würden eine Klausur nur mit einer einzigen Aufgabe bewerten. Könnte das Ergebnis zufällig besonders gut oder besonders schlecht ausfallen?

Das Problem liegt auf der Hand.

Train/Test-Split: Einmalige Aufteilung der Daten in Trainings- und Testmenge, typischerweise 80/20 oder 70/30.

Ein einzelner Split ist wie eine Klausur mit nur einer Aufgabe: Hat der Student gerade diese Aufgabe gut geübt, sieht sein Ergebnis brillant aus; war es sein schwacher Bereich, fällt er durch. Weder das eine noch das andere Ergebnis repräsentiert sein wahres Können. Genau so verhält es sich mit dem Train/Test-Split. Der Zufall bestimmt, welche Datenpunkte im Testset landen, und dieser Zufall kann die geschätzte Performance nach oben oder unten verzerren.

5.2 Die Analogie der rotierenden Richter

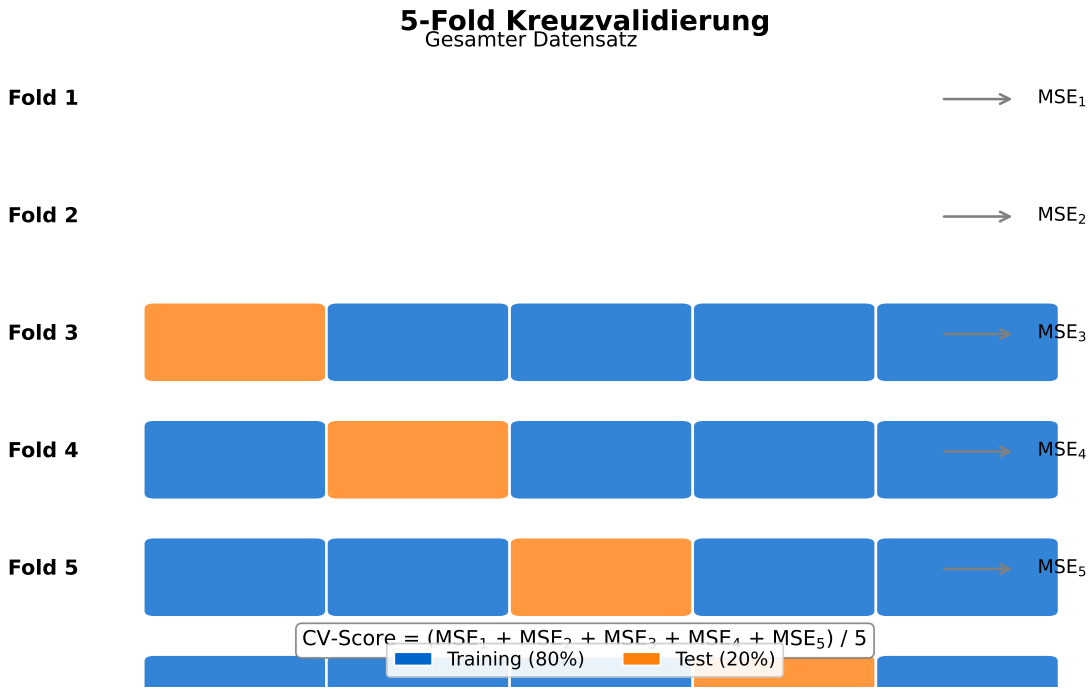
Stellen wir uns ein Gericht vor, in dem fünf Richter

K-Fold CV: Verfahren, bei dem die Daten in K gleich große Teile (Folds) aufgeteilt werden. Jeder Fold dient einmal als Testset, die übrigen $K-1$ Folds als Trainingsset.

sitzen. In der ersten Runde urteilt Richter 1, während die Richter 2 bis 5 beobachten. In der zweiten Runde urteilt Richter 2, die anderen schauen zu. So rotiert die Rolle des Richters, bis jeder einmal geurteilt hat. Am Ende mittelt man die fünf Urteile.

Das Ergebnis ist fairer als ein einzelnes Urteil, denn die persönliche Tendenz eines einzelnen Richters mittelt sich heraus. In der Sprache der Statistik: Die Varianz der Performanceschätzung sinkt, weil wir über mehrere unabhängige "Blickwinkel" mitteln.

Übertragen auf Lenas Modell: Jeder Fold (Datenteil) übernimmt einmal die Rolle des Richters (Testset), während die restlichen Folds trainieren. Am Ende hat *jeder einzelne Datenpunkt* genau einmal als Testpunkt gedient. Kein Punkt wird doppelt getestet, keiner wird vergessen.



5.3 K-Fold Cross-Validation Schritt für Schritt

K-Fold Cross-Validation

Gegeben ein Datensatz mit n Beobachtungen und eine Wahl von K (typisch $K = 5$ oder $K = 10$):

1. Mische die Daten zufällig und teile sie in K gleich große, nicht überlappende Teilmengen (*Folds*) auf.
2. Für $k = 1, 2, \ldots, K$: Trainiere das Modell auf den Folds $\{1, \ldots, K\} \setminus \{k\}$ und evaluiere es auf Fold k. Speichere den Testfehler e_k.
3. Berechne den Cross-Validation-Fehler als Durchschnitt:

$$\mathrm{CV}(K) \;=\; \frac{1}{K} \sum_{k=1}^{K} e_k$$

Fold: Einer der K Teilmengen, in die der Datensatz aufgeteilt wird. Jeder Fold enthält n/K Beobachtungen (ggf. gerundet).

Warum funktioniert dieses Vorgehen? Weil jeder Datenpunkt *genau einmal* im Testset landet. Beim einfachen Train/Test-Split bleiben 20 % der Daten für immer im Testset und 80 % für immer im Training. Bei 5-Fold CV hingegen wechseln die Rollen: Jeder der fünf Blöcke ist einmal Testset und viermal Trainingsset. Dadurch wird die Schätzung robuster, weil sie nicht von einem einzelnen glücklichen oder unglücklichen Split abhängt.

Für Lenas DataCo-Datensatz mit 500 Kunden ergibt sich bei $K = 5$: Jeder Fold enthält 100 Kunden. In jedem Durchlauf trainiert das Modell auf 400 Kunden und wird an den verbleibenden 100 getestet. Nach fünf Durchläufen hat Lena fünf Accuracy-Werte, deren Mittelwert eine deutlich stabilere Schätzung liefert als ein einzelner Split.

CV-Fehler und Standardabweichung

$$\widehat{\mathrm{CV}} = \frac{1}{K}\sum_{k=1}^{K} e_k, \qquad \mathrm{SD}_{\mathrm{CV}} = \sqrt{\frac{1}{K-1}\sum_{k=1}^{K}(e_k - \widehat{\mathrm{CV}})^2}$$

Symbol für Symbol:

- K – Anzahl der Folds (z. B. 5 oder 10)
- e_k – Testfehler (z. B. Accuracy, MSE oder AUC) auf Fold k
- $\widehat{\mathrm{CV}}$ – gemittelter CV-Fehler über alle Folds
- $\mathrm{SD}_{\mathrm{CV}}$ – Standardabweichung der Fold-Fehler; misst, wie stabil die Schätzung ist

Eine kleine $\mathrm{SD}_{\mathrm{CV}}$ bedeutet, dass das Modell auf allen Folds ähnlich gut abschneidet – ein Zeichen für Robustheit.

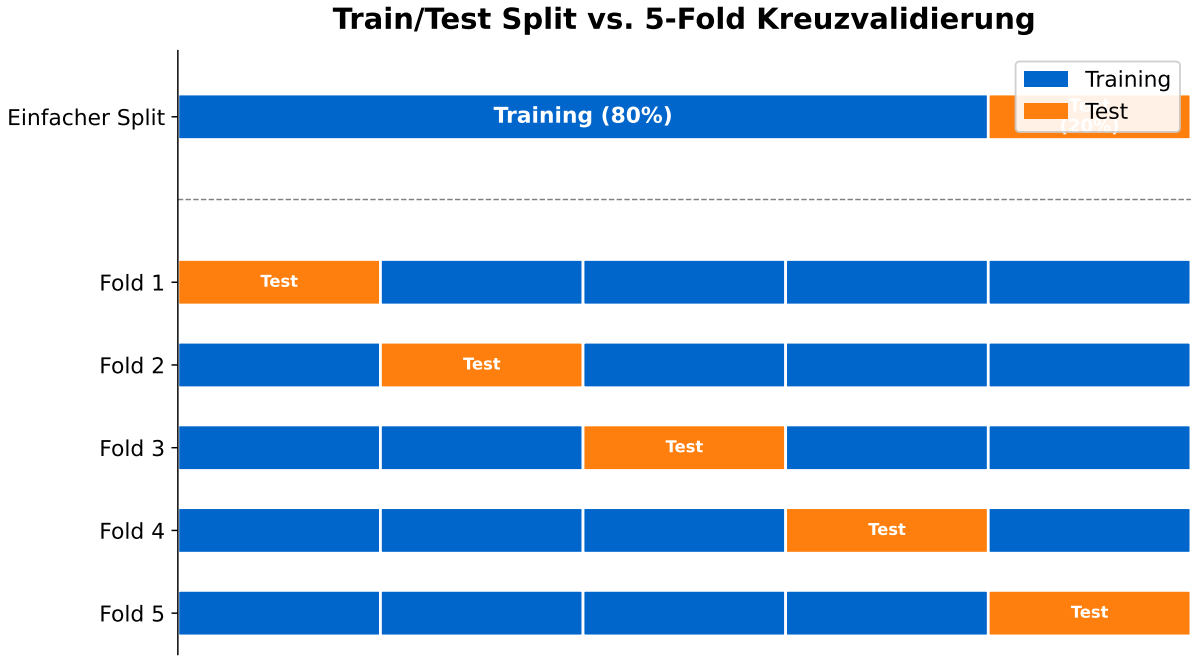


Abbildung 13: Visualisierung eines Train/Test-Splits und K-Fold Rotation

5.4 CV-Varianten

Nicht jede Situation verlangt dasselbe CV-Schema. Die folgende Tabelle fasst die wichtigsten Varianten zusammen; danach erklären wir jede einzeln.

Variante	K	**Bias/Varianz**	**Wann verwenden?**
K-Fold	5 oder 10	Guter Kompromiss	Standardwahl
LOOCV	n	Niedrig / Hoch	Sehr kleine Datensätze
Stratified K-Fold	5 oder 10	Wie K-Fold	Unbalancierte Klassen
Nested CV	aussen + innen	Unverzerrt	Hyperparameter-Tuning
Time Series CV	variabel	Kontextabhängig	Zeitlich geordnete Daten

K-Fold ($K = 5$ oder $K = 10$)

K-Fold CV: Standardverfahren mit $K = 5$ oder $K = 10$. Empirische Studien zeigen, dass $K = 10$ den besten Kompromiss zwischen Bias und Varianz der CV-Schätzung liefert.

Die Standard-Kreuzvalidierung mit fünf oder zehn Folds. Empirische Vergleiche von Breiman & Spector (1992) und Kohavi (1995) zeigen, dass $K = 10$ in den meisten Situationen den besten Kompromiss liefert: Der Bias ist gering, weil 90 % der Daten zum Training genutzt werden, und die Varianz bleibt handhabbar, weil zehn Folds genügend unabhängige Perspektiven bieten.

Lena wählt$K = 5$, weil ihr Datensatz mit 500 Kunden eher klein ist und $K = 5$ schneller zu berechnen ist. Der Unterschied zu $K = 10$ ist in der Praxis oft marginal.

LOOCV – Leave-One-Out Cross-Validation ($K = n$)

LOOCV: Spezialfall von K-Fold mit $K = n$. Jeder einzelne Datenpunkt dient einmal als Testset, das Modell wird auf $n-1$ Beobachtungen trainiert.

Im Extremfall setzt man $K = n$: In jeder Runde wird genau ein Datenpunkt herausgenommen und getestet, während $n-1$ Punkte trainieren. Vorteil: Der Bias ist minimal, weil fast der gesamte Datensatz zum Training genutzt wird. Nachteil: Die Varianz der Schätzung kann hoch sein, weil die n Trainingsmengen sich nur um einen einzigen Punkt unterscheiden – die Modelle sind fast identisch, und ihre Fehler korrelieren stark.

Zusätzlich ist LOOCV rechenintensiv: Für Lenas 500 Kunden müsste sie das Modell 500 Mal trainieren. Bei linearer Regression existiert eine geschlossene Formel (über die Hat-Matrix), aber für komplexere Modelle ist LOOCV oft zu teuer.

Stratified K-Fold

Stratified K-Fold: CV-Variante, bei der die Klassenverteilung in jedem Fold der Gesamtverteilung entspricht. Unverzichtbar bei unbalancierten Daten.

Bei Lenas Churn-Daten sind nur 15 % der Kunden Kündiger. Ohne Stratifizierung könnte es passieren, dass ein Fold zufällig fast keine Kündiger enthält – die Evaluation auf diesem Fold wäre wenig aussagekräftig. Stratified K-Fold erzwingt, dass die Klassenverteilung in jedem Fold die Gesamtverteilung widerspiegelt. In Lenas Fall enthält jeder Fold genau 15 % Kündiger.

In der Praxis verwenden alle modernen Softwarepakete (z. B. `caret` in R oder `scikit-learn` in Python) standardmäßig Stratified K-Fold für Klassifikationsprobleme.

Nested Cross-Validation

Nested CV: Doppelte Schleife: Die äußere Schleife schätzt die Modellperformance, die innere Schleife wähltdie besten Hyperparameter. Verhindert optimistischen Bias.

Wenn Lena nicht nur die Performance schätzen, sondern gleichzeitig Hyperparameter tunen will (z. B. λ bei Lasso), braucht sie eine doppelte Schleife. Die äußere Schleife (z. B. $K_{\text{aussen}} = 5$) schätzt die finale Performance, die innere Schleife (z. B. $K_{\text{innen}} = 5$) wähltfür jeden äußeren Fold die besten Hyperparameter.

Ohne Nested CV würde Lena die Hyperparameter auf denselben Daten wählen, auf denen sie auch die Performance bewertet – das führt zu einem optimistischen Bias, weil die Modellselektion bereits Information über die Testdaten "gesehen" hat.

Schematisch:

1. **Äußere Schleife** (K_{aussen} Folds): Teile Daten in äußeren Trainings- und Testteil.
2. **Innere Schleife** (K_{innen} Folds): Führe auf dem äußeren Trainingsteil eine CV durch, um den besten Hyperparameter zu wählen.
3. Trainiere das finale Modell mit dem besten Hyperparameter auf dem gesamten äußeren Trainingsteil und evaluiere auf dem äußeren Testteil.

4. Wiederhole für alle äußeren Folds und mittele.

Time Series Cross-Validation

Time Series CV: CV-Verfahren, bei dem nur vergangene Daten zum Training und zukünftige zum Testen verwendet werden. Respektiert die zeitliche Ordnung.

Bei zeitlich geordneten Daten – etwa monatlichen Umsatzzahlen oder täglichen Churn-Raten – ist eine zufällige Fold-Zuteilung verboten. Der Grund: Wenn ein Modell auf Daten von Mai trainiert und auf Daten von April getestet wird, "sieht" es die Zukunft. Das nennt man *Data Leakage*.

Data Leakage: Verwendung von Informationen im Training, die zur Testzeit nicht verfügbar wären

Time Series CV respektiert die Zeitachse: In jedem Durchlauf wird auf einem wachsenden Fenster vergangener Daten trainiert und auf dem nächsten Zeitabschnitt getestet. So simuliert man die reale Situation, in der Prognosen immer nur für die Zukunft gelten.

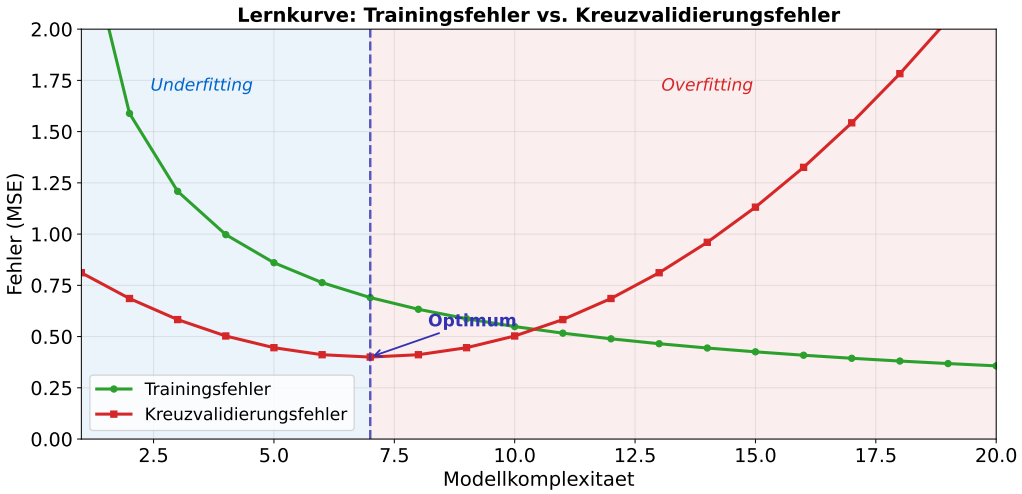


Abbildung 14: Lernkurve: Training- und CV-Fehler als Funktion der Modellkomplexität

5.5 CV und Regularisierung: Das Dream-Team

Regularisierung hat einen Hyperparameter λ, der die Stärke der Schrumpfung steuert (siehe Abschnitt 4). Dieser Parameter muss *auf den Daten* gewählt werden – und genau dafür ist Kreuzvalidierung unersetzlich.

`cv.glmnet`: Funktion aus dem R-Paket `glmnet`, die automatisch K-Fold CV durchführt und das optimale λ wählt.

Die Funktion `cv.glmnet` in R kombiniert beides in einer eleganten Pipeline:

1. Erzeuge ein Raster von λ-Werten (typischerweise 100 Werte auf einer logarithmischen Skala).
2. Für jedes λ: Führe K-Fold CV durch und berechne den mittleren CV-Fehler.
3. Wähle das λ mit dem kleinsten CV-Fehler (`lambda.min`).
4. Optional: Wähle das sparsamste Modell, dessen CV-Fehler innerhalb einer Standardabweichung des Minimums liegt (`lambda.1se`).

Lenas vollständige Pipeline sieht damit so aus:

`cv.glmnet` (innere CV) $\longrightarrow$ optimales λ $\longrightarrow$ finales Modell $\longrightarrow$ Evaluation auf Holdout-Set

Wichtig: Das finale Holdout-Set war zu keinem Zeitpunkt an der Modellselektion oder am Hyperparameter-Tuning beteiligt. Es dient ausschließlich der ehrlichen Leistungsbewertung am Ende.

5.6 Beispiel mit Zahlen

Lenas 5-Fold CV bei DataCo

Lena teilt ihre 500 Kunden in fünf Folds auf und trainiert ein logistisches Regressionsmodell (mit Lasso-Regularisierung) auf jeweils vier Folds:

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mittel
Accuracy	0.74	0.78	0.77	0.73	0.78	0.760
AUC	0.81	0.84	0.83	0.80	0.82	0.820

Ergebnis: Mean Accuracy $= 0.76$ (SD $= 0.024$), Mean AUC $= 0.82$ (SD $= 0.015$).
Die niedrige Standardabweichung zeigt, dass das Modell stabil über alle Folds performt. Lena kann ihrem Chef mit hoher Zuversicht berichten, dass die Accuracy bei etwa 76 % liegt – nicht bei den 78 % des einen glücklichen Splits und nicht bei den 71 % des unglücklichen.

Lena vergleicht außerdem die CV-Ergebnisse verschiedener Modelle:

Modell	**CV-Accuracy**	**CV-AUC**
OLS (alle Features)	0.72 (0.04)	0.78 (0.03)
Lasso ($\lambda_{\min}$)	0.76 (0.02)	0.82 (0.02)
Ridge ($\lambda_{\min}$)	0.75 (0.03)	0.81 (0.02)

Lasso gewinnt: höhere Accuracy, höhere AUC, und die kleinste Streuung. Die automatische Feature-Selektion des Lasso hat drei irrelevante Prädiktoren eliminiert und damit die Varianz des Modells reduziert.

Die Geburt der Kreuzvalidierung

1974 – Mervyn Stone veröffentlicht den Artikel "Cross-Validatory Choice and Assessment of Statistical Predictions". Er formalisiert die Idee, Daten systematisch aufzuteilen, um die Vorhersagegüte eines Modells zu schätzen, ohne neue Daten sammeln zu müssen. Stone erkennt, dass ein Modell, das auf Trainingsdaten gut aussieht, bei neuen Daten versagen kann – ein Problem, das heute unter dem Namen *Overfitting* allgegenwärtig ist.
1975 – Seymour Geisser formalisiert das Konzept unter dem Namen "predictive sample reuse" und legt die theoretischen Grundlagen für LOOCV.
1992 – Leo Breiman und Richard Spector führen eine umfangreiche empirische Studie durch, in der sie verschiedene CV-Varianten vergleichen. Ihr Ergebnis: $K = 10$ liefert in den meisten Fällen den besten Kompromiss.
1995 – Ron Kohavi publiziert die vielleicht einflussreichste Vergleichsstudie. Er zeigt, dass 10-Fold CV dem Bootstrap und LOOCV bei der Modellselektion überlegen ist und empfiehlt *Stratified 10-Fold* als allgemeinen Standard. Diese Empfehlung hat sich bis heute gehalten.

Häufige Fehler

Fehler 1: "LOOCV ist immer am besten, weil der Bias minimal ist."
Das stimmt für den Bias, aber nicht für die Gesamtqualität der Schätzung. LOOCV hat eine hohe Varianz, weil die n Trainingsmengen fast identisch sind und ihre Fehler stark korrelieren. Außerdem ist LOOCV bei großen Datensätzen oder komplexen Modellen rechnerisch teuer. In der Praxis ist 10-Fold CV fast immer die bessere Wahl.
Fehler 2: "CV ersetzt ein Holdout-Testset."
Nein. Kreuzvalidierung ist ein Werkzeug für die *Modellselektion*: Welches Modell ist am besten? Welcher Hyperparameter ist optimal? Aber die finale, ehrliche Leistungsbewertung sollte auf einem separaten Holdout-Set erfolgen, das zu keinem Zeitpunkt für Training oder Modellauswahl verwendet wurde. Sonst besteht die Gefahr, dass man das Modell unbewusst auf die CV-Folds überanpasst.
Fehler 3: "Mehr Folds sind immer besser."
Der Grenznutzen zusätzlicher Folds nimmt ab. Zwischen $K = 5$ und $K = 10$ ist der Unterschied spürbar, zwischen $K = 10$ und $K = 20$ kaum noch. Gleichzeitig steigt der Rechenaufwand linear mit K. Ein guter Kompromiss ist $K = 10$, in zeitkritischen Situationen $K = 5$.

5.7 R-Code: caret und cv.glmnet

5-Fold CV mit caret

```r
library(caret)

# CV-Steuerung: 5-Fold, stratifiziert
ctrl <- trainControl(
  method      = "cv",
  number      = 5,
  classProbs  = TRUE,
  summaryFunction = twoClassSummary
)

# Logistisches Modell mit CV
model <- train(
  churn ~ ., data = dataco,
  method      = "glm",
  family      = "binomial",
  trControl   = ctrl,
  metric      = "ROC"
)

print(model)
# => ROC   0.820   Sens   0.68   Spec   0.84
```

Lasso mit cv.glmnet

```r
library(glmnet)

x <- model.matrix(churn ~ ., data = dataco)[, -1]
y <- dataco$churn

# Kreuzvalidiertes Lasso (alpha = 1)
cv_fit <- cv.glmnet(x, y, alpha = 1,
                    family = "binomial", nfolds = 5)

# Optimales Lambda
cat("lambda.min:", cv_fit$lambda.min)    # z.B. 0.023
cat("lambda.1se:", cv_fit$lambda.1se)    # z.B. 0.041

# Koeffizienten: 3 von 12 auf Null gesetzt
coef(cv_fit, s = "lambda.min")
```

Manuelle K-Fold-Implementierung

```r
set.seed(42)
K <- 5
folds <- sample(rep(1:K, length.out = nrow(dataco)))
errors <- numeric(K)

for (k in 1:K) {
  train_k <- dataco[folds != k, ]
  test_k  <- dataco[folds == k, ]

  fit_k <- glm(churn ~ ., data = train_k,
               family = binomial)
  pred_k <- predict(fit_k, test_k, type = "response")
  class_k <- ifelse(pred_k > 0.5, 1, 0)

  errors[k] <- mean(class_k == test_k$churn)
}

cat("Mean Accuracy:", mean(errors), "\n")
cat("SD Accuracy:  ", sd(errors), "\n")
```

5.8 Übungsaufgaben

Aufgabe 5.1 – Fold-Größen

Lena hat $n = 120$ Datenpunkte und wählt$K = 10$.

(a) Wie viele Datenpunkte enthält jeder Fold?

(b) Wie viele Datenpunkte werden in jedem Durchlauf zum Training verwendet?

(c) Wie oft wird jeder einzelne Datenpunkt insgesamt zum Training herangezogen?

Aufgabe 5.2 – LOOCV vs. 10-Fold

Erklären Sie in eigenen Worten, warum LOOCV trotz minimalen Bias nicht immer besser ist als 10-Fold CV. Gehen Sie auf Varianz und Rechenaufwand ein.

Aufgabe 5.3 – Stratifizierung

Lenas DataCo-Datensatz enthält 85 % Nicht-Kündiger und 15 % Kündiger.

(a) Was könnte passieren, wenn sie bei $K = 5$ *ohne* Stratifizierung aufteilt?

(b) Wie stellt Stratified K-Fold sicher, dass jeder Fold repräsentativ ist?

Aufgabe 5.4 – Nested CV

Erklären Sie, warum es zu einem optimistischen Bias führt, wenn man Hyperparameter (z. B. λ) auf denselben Daten wählt, auf denen man auch die Performance bewertet. Skizzieren Sie die Lösung mit Nested CV.

Aufgabe 5.5 – CV-Ergebnisse interpretieren

Lena erhält für zwei Modelle folgende 5-Fold-CV-Ergebnisse:

Modell	**Mean Accuracy**	**SD**
Modell A	0.80	0.08
Modell B	0.77	0.02

Welches Modell würden Sie empfehlen und warum? Berücksichtigen Sie sowohl die mittlere Accuracy als auch die Streuung.

Kernaussage: Der CV-Fehler ist die beste Schätzung für die Out-of-Sample Performance eines Modells. Kreuzvalidierung stellt sicher, dass jeder Datenpunkt genau einmal im Testset liegt, und liefert damit eine stabilere und fairere Bewertung als ein einzelner Train/Test-Split.

6 Verursacht mein Regenschirm den Regen? – Kausalität

Lenas Herausforderung

Lena prasentiert ihrem Chef eine Korrelation von $r = 0{,}45$ zwischen Marketing-Ausgaben und Kundenbindung. Der Chef ist begeistert und will das Marketing-Budget sofort verdreifachen. Aber Lena zögert: Bedeutet die Korrelation wirklich, dass *Marketing die Bindung verursacht*? Oder steckt etwas anderes dahinter?

6.1 Entdeckungsfrage

Entdeckungsfrage

Eiscremeverkauf und Ertrinkungsfälle steigen beide im Sommer. Sollen wir Eiscreme verbieten, um Menschenleben zu retten?

Natürlich nicht. Die Hitze im Sommer treibt beides: mehr Eisverkauf *und* mehr Schwimmbadbesuche (und damit mehr Ertrinkungsfälle). Eiscreme verursacht kein Ertrinken. Aber die Korrelation ist real – sie entsteht durch eine dritte Variable, die auf beide einwirkt.

Dieses Muster begegnet uns überall: im Alltag, in der Wissenschaft und in Lenas DataCo-Daten. Die zentrale Frage dieses Abschnitts lautet: Wie unterscheiden wir echte Ursache-Wirkungs-Beziehungen von bloßen statistischen Zusammenhängen?

6.2 Regenschirm und Puppenspieler

Korrelation: Statistischer Zusammenhang zwischen zwei Variablen. Sagt nichts über die Richtung oder Ursache des Zusammenhangs aus.

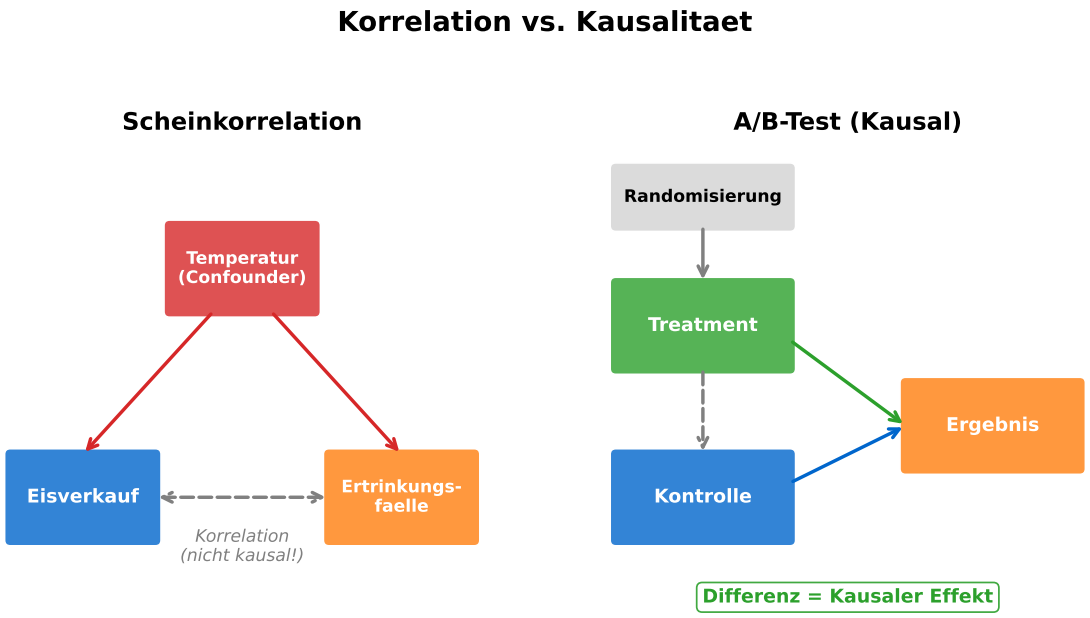
Stellen Sie sich vor, jemand beobachtet: “Jedes Mal, wenn ich meinen Regenschirm mitnehme, regnet es.” Verursacht der Regenschirm den Regen? Offensichtlich nicht. Der wahre Mechanismus ist: Die Wettervorhersage sagt Regen voraus, also nehmen wir den Schirm mit, und es regnet tatsächlich. Der Schirm und der Regen haben eine gemeinsame Ursache (die Wetterlage), aber keiner verursacht den anderen.

Confounder: Drittvariable, die sowohl die vermeintliche Ursache als auch die vermeintliche Wirkung beeinflusst und so eine Scheinkorrelation erzeugt.

Noch eindrucksvoller ist die Analogie des Puppenspielers. Stellen Sie sich eine Marionetten-Buehne vor: Zwei Puppen (Variable X und Variable Y) scheinen miteinander zu interagieren. Aber hinter dem Vorhang zieht ein Puppenspieler (der Confounder) an den Fäden beider Puppen. Wer nur die Buehne sieht, glaubt, X beeinflusse Y. Wer hinter den Vorhang schaut, erkennt, dass eine dritte Kraft beide steuert.

Für Lena bei DataCo: Marketing-Ausgaben (X) und Kundenbindung (Y) korrelieren. Aber hinter dem Vorhang zieht vielleicht die *Konjunktur* an beiden Fäden: In guten Zeiten investiert DataCo mehr in Marketing *und* Kunden bleiben länger (weil es ihnen finanziell gut geht). Die Korrelation zwischen Marketing und Bindung wäre dann großteils eine Scheinkorrelation – erzeugt durch den Puppenspieler “Konjunktur”.

6.3 Korrelation vs. Kausalität



Kausalität: X verursacht Y, wenn eine Änderung in X – bei Konstanthaltung aller anderen Faktoren – zu einer Änderung in Y führt.

Wenn zwei Variablen korrelieren, gibt es genau drei mögliche Erklärungen:

Drei Mechanismen hinter Korrelation

1. **Confounding (Drittvariable):** Eine versteckte Variable Z beeinflusst sowohl X als auch Y. Die Korrelation zwischen X und Y ist eine Scheinkorrelation, die verschwindet, sobald man Z kontrolliert.

2. **Reverse Causation (Umgekehrte Kausalität):** Nicht X verursacht Y, sondern Y verursacht X. Beispiel: Kranke Menschen nehmen mehr Medikamente – die Medikamente verursachen nicht die Krankheit, sondern die Krankheit verursacht die Medikamenteneinnahme.

3. **Zufall (Spurious Correlation):** Bei genug Variablen und genug Tests findet man immer Korrelationen, die rein zufällig sind. Tyler Vigen hat hunderte solcher Beispiele gesammelt: Die Scheidungsrate in Maine korreliert $r = 0{,}99$ mit dem Pro-Kopf-Margarineverbrauch.

In keinem dieser drei Fälle würde eine Intervention an X eine Änderung in Y bewirken. Deshalb ist die Unterscheidung zwischen Korrelation und Kausalität nicht bloss akademisch – sie hat direkte Konsequenzen für Entscheidungen. Wenn Lenas Chef das Marketing-Budget verdreifacht, aber die Korrelation durch die Konjunktur getrieben war, wird er enttäuscht werden.

6.4 Fünf Kriterien für Kausalität

Wann dürfen wir von einer kausalen Beziehung sprechen? Die folgende Liste destilliert die klassischen Kriterien (angelehnt an Bradford Hill, 1965) auf fünf praxistaugliche Punkte:

Fünf Kriterien für Kausalität

1. **Korrelation vorhanden:** X und Y müssen statistisch zusammenhängen. Ohne Korrelation keine Kausalität (aber Korrelation allein reicht nicht).
2. **Zeitliche Reihenfolge:** Die Ursache muss *vor* der Wirkung eintreten. Marketing muss vor der Bindungsmessung stattfinden, nicht danach.
3. **Kein Confounding:** Alle relevanten Drittvariablen müssen kontrolliert oder ausgeschlossen sein. Das ist die härteste Bedingung, denn man kann nicht kontrollieren, was man nicht kennt.
4. **Plausibler Mechanismus:** Es sollte eine nachvollziehbare Erklärung geben, *warum* X auf Y wirkt. "Margarine verursacht Scheidungen" hat keinen plausiblen Mechanismus.
5. **Konsistenz über Studien:** Der Zusammenhang zeigt sich wiederholt in verschiedenen Kontexten, Stichproben und Ländern.

Hill-Kriterien: 1965 von Sir Austin Bradford Hill formulierte neun Kriterien für Kausalität aus Beobachtungsdaten (hier auf fünf Kernpunkte verdichtet).

Der absolute Goldstandard bleibt das **randomisierte Experiment** (A/B-Test). Warum? Weil die zufällige Zuteilung *alle* Confounders eliminiert – auch solche, die wir nicht kennen oder nicht messen können. Die fünf Kriterien helfen dann, wenn ein Experiment nicht möglich ist (z. B. aus ethischen oder praktischen Gründen).

6.5 Pearls Leiter der Kausalität

Judea Pearl: Informatiker an der UCLA, Turing Award 2011. Begründer der modernen kausalen Inferenz mit dem *do*-Kalkül und gerichteten azyklischen Graphen (DAGs).

Der Informatiker Judea Pearl hat eine elegante Hierarchie vorgeschlagen, die drei Stufen des kausalen Denkens unterscheidet. Diese "Leiter der Kausalität" hilft, die eigene Analyse einzuordnen.

Pearls Leiter der Kausalität (vereinfacht)

Stufe 1 – Sehen (Assoziation):
"Kunden, die Marketing-Mails öffnen, kündigen seltener."
Das ist eine reine Beobachtung. Wir beschreiben, was *zusammen auftritt*.

Stufe 2 – Tun (Intervention):
"Wenn wir mehr Marketing-Mails senden, kündigen die Kunden seltener."
Das ist eine kausale Aussage. Wir verändern aktiv X und beobachten die Wirkung auf Y. Der mathematische Operator ist Pearls $\text{do}(X = x)$.

Stufe 3 – Vorstellen (Kontrafaktisch):
"Hätte dieser spezifische Kunde gekündigt, wenn wir ihm *keine* Marketing-Mail geschickt hätten?"
Das ist die anspruchsvollste Stufe. Wir fragen nach einer alternativen Welt, die nie stattgefunden hat.

Jede Stufe baut auf der vorherigen auf und erfordert stärkere Annahmen. Lenas Korrelationsanalyse ($r = 0{,}45$) bewegt sich auf Stufe 1. Um auf Stufe 2 zu gelangen, bräuchte sie ein Experiment. Stufe 3 erfordert zusätzlich ein kausales Modell, das kontrafaktische Fragen beantworten kann.

Pearls do-Operator (vereinfacht)

Der Unterschied zwischen Beobachtung und Intervention lässt sich formal ausdrücken:

$$\underbrace{P(Y \mid X = x)}_{\text{Stufe 1: Beobachtet}} \neq \underbrace{P(Y \mid \text{do}(X = x))}_{\text{Stufe 2: Interveniert}}$$

Symbol für Symbol:

- $P(Y \mid X = x)$ – Wahrscheinlichkeit von Y, *gegeben dass wir beobachten* $X = x$
- $P(Y \mid \text{do}(X = x))$ – Wahrscheinlichkeit von Y, *wenn wir aktiv setzen* $X = x$
- $\text{do}(X = x)$ – der Eingriff, der X auf den Wert x fixiert und alle kausalen Einflüsse *auf* X kappt

In der Beobachtung können Confounders die Beziehung verzerren. Durch den do-Operator werden diese Verzerrungen eliminiert – ähnlich wie bei einem randomisierten Experiment.

6.6 Confounding erklärt

Confounding ist das zentrale Problem der kausalen Inferenz aus Beobachtungsdaten. Wir haben es bereits informell kennengelernt; jetzt vertiefen wir es.

Confounder (formale Definition)

Eine Variable Z ist ein Confounder für die Beziehung zwischen X und Y, wenn:

1. Z beeinflusst X (kausaler Pfad $Z \to X$),
2. Z beeinflusst Y (kausaler Pfad $Z \to Y$),
3. Z liegt nicht auf dem kausalen Pfad von X nach Y.

Dadurch erzeugt Z eine Scheinassoziation zwischen X und Y, die verschwindet, sobald man Z kontrolliert.

Eiscreme und Ertrinken

Scheinkorrelation: Korrelation zwischen X und Y, die vollständig durch einen Confounder Z erklärt wird und verschwindet, sobald man Z kontrolliert.

Im Sommer steigt die Temperatur. Das führt zu mehr Eisverkauf *und* zu mehr Schwimmbadbesuchen (und damit zu mehr Ertrinkungsfällen). Die Temperatur ist der Confounder. Sobald wir die Temperatur kontrollieren (z. B. nur Tage mit 25°C vergleichen), verschwindet die Korrelation zwischen Eisverkauf und Ertrinken.

Lenas Konjunktur-Confounder

In guten Konjunkturzeiten investiert DataCo mehr in Marketing (weil Budget vorhanden ist), und gleichzeitig bleiben Kunden länger (weil sie weniger preissensibel sind). Die Konjunktur beeinflusst beides. Die beobachtete Korrelation von $r = 0{,}45$ zwischen Marketing und Kundenbindung könnte großteils oder vollständig durch die Konjunktur getrieben sein.

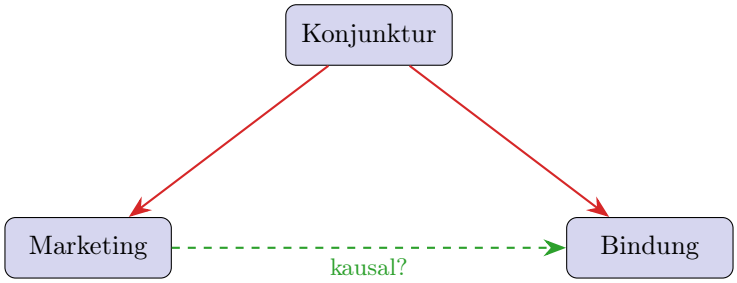
6.7 Gerichtete azyklische Graphen (DAGs)

DAG: Directed Acyclic Graph. Ein Diagramm mit Knoten (Variablen) und gerichteten Pfeilen (kausale Wirkungen). "Azyklisch" bedeutet: Kein Pfeil führt im Kreis zurück.

Ein DAG (Directed Acyclic Graph) ist das Standardwerkzeug, um kausale Beziehungen visuell darzustellen. Die Regeln sind einfach:

- Jeder **Knoten** repräsentiert eine Variable.
- Jeder **Pfeil** repräsentiert eine direkte kausale Wirkung (von Ursache zu Wirkung).
- **Azyklisch** bedeutet: Man kann den Pfeilen nicht im Kreis folgen.

Lenas DataCo-Situation als DAG:



Die roten Pfeile zeigen die gesicherten Wirkungen: Konjunktur beeinflusst sowohl Marketing als auch Bindung. Der grüne gestrichelte Pfeil ist die offene Frage: Gibt es *zusätzlich* einen direkten kausalen Effekt von Marketing auf Bindung? Um das herauszufinden, muss Lena den Confounder "Konjunktur" kontrollieren.

Confounder-Struktur: Kausales DAG-Beispiel

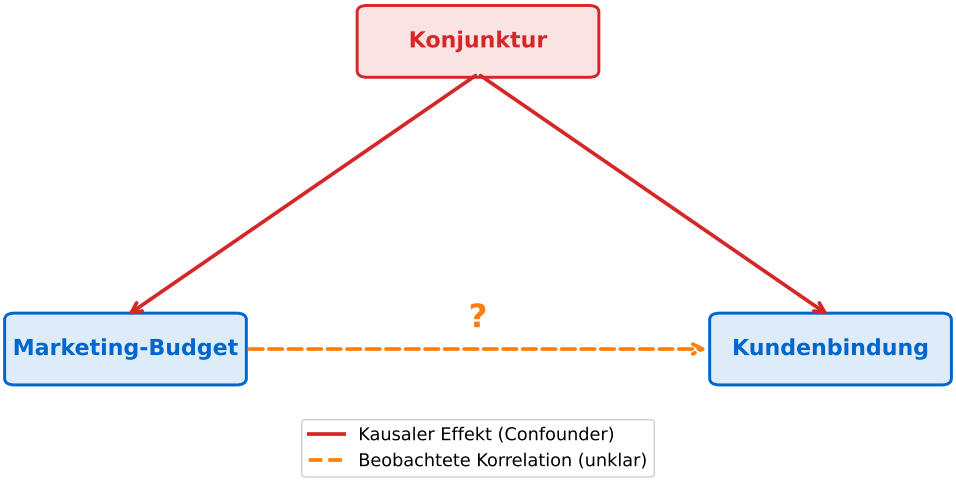


Abbildung 15: Gerichteter azyklischer Graph (DAG): Konjunktur als Confounder

6.8 Methoden der kausalen Inferenz (Überblick)

Wenn ein randomisiertes Experiment nicht möglich ist, gibt es fortgeschrittene statistische Methoden, die helfen können, kausale Effekte aus Beobachtungsdaten zu schätzen. Wir geben hier einen kurzen Überblick auf BSc-Niveau – jede dieser Methoden ist ein eigenes Semester wert.

Methode	Kernidee	Annahme
Kontrollvariablen	Confounder ins Modell aufnehmen	Alle Confounder bekannt
Matching	Vergleichbare Paare bilden	Gute Matches möglich
Instrumentvariablen (IV)	Exogene Variation nutzen	Gültiges Instrument existiert
Difference-in-Differences	Vorher-Nachher-Vergleich	Parallele Trends

Kontrollvariablen

Der einfachste Ansatz: Den Confounder als zusätzliche Variable ins Regressionsmodell aufnehmen. Wenn Lena "Konjunkturindex" als Prädiktor hinzufügt, kontrolliert das Modell für die Konjunktur, und der verbleibende Effekt von Marketing auf Bindung ist (hoffentlich) näher am kausalen Effekt.

Problem: Das funktioniert nur für *bekannte* Confounders. Unbekannte oder nicht gemessene Confounders können weiterhin verzerren. Die kritische Annahme heißt "no unmeasured confounding" – und sie ist in der Praxis selten garantiert.

Matching

Statt den Confounder statistisch zu kontrollieren, bildet man Paare: Für jeden Kunden mit hohen Marketing-Kontakten sucht man einen "Zwilling" mit niedrigen Marketing-Kontakten, der aber in allen Confoundern (Alter, Einkommen, Region) ähnlich ist. Die Differenz in der Bindung zwischen den Zwillingen schätzt den kausalen Effekt.

Instrumentvariablen (IV)

Manchmal gibt es eine Variable, die X beeinflusst, aber keinen *direkten* Effekt auf Y hat (ausser über X). Diese Variable heißt Instrument. Beispiel: Ein Werbegutschein, der zufällig verteilt wird, beeinflusst die Marketing-Exposition, hat aber keinen direkten Effekt auf die Kundenbindung (ausser über das Marketing).

Difference-in-Differences (DiD)

Vergleiche eine Gruppe vor und nach einer Intervention mit einer Kontrollgruppe, die die Intervention nicht erhalten hat. Die Differenz der Differenzen schätzt den kausalen Effekt, unter der Annahme, dass beide Gruppen ohne Intervention den gleichen Trend gehabt hätten ("Parallel Trends").

6.9 Beispiel: Lenas Marketing-Puzzle

Lenas Marketing-Puzzle bei DataCo

Schritt 1 – Rohe Korrelation:
Lena berechnet die Korrelation zwischen monatlichen Marketing-Ausgaben und 30-Tage-Retention über 24 Monate: $r = 0{,}45$, $p < 0{,}05$. Der Chef ist begeistert.
Schritt 2 – Confounder identifizieren:
Lena überlegt: Was könnte beides beeinflussen? Sie hat Zugang zum Konjunkturindex (BIP-Wachstum). Tatsächlich korreliert der Konjunkturindex sowohl mit Marketing ($r = 0{,}62$) als auch mit Retention ($r = 0{,}58$).
Schritt 3 – Partielle Korrelation:
Lena berechnet die partielle Korrelation zwischen Marketing und Retention, kontrolliert für den Konjunkturindex:

$$r_{\text{Marketing, Retention|Konjunktur}} = 0{,}12 \quad (p = 0{,}38,\ \text{n.s.})$$

Ergebnis: Nach Kontrolle für die Konjunktur ist der Zusammenhang zwischen Marketing und Retention klein und statistisch nicht signifikant. Die ursprüngliche Korrelation von $r = 0{,}45$ war großteils eine Scheinkorrelation, getrieben durch den Confounder "Konjunktur".
Lenas Fazit an den Chef:
"Die Daten zeigen keine überzeugenden Belege dafür, dass Marketing die Bindung direkt verbessert. Um das sicher zu wissen, brauchen wir einen A/B-Test."

Drei Geschichten über Kausalität

Judea Pearl und der Turing Award (2011)
Judea Pearl, geboren 1936 in Tel Aviv, kam als Informatiker zur Künstlichen Intelligenz und stellte fest, dass Maschinen nicht "verstehen" können, was Ursache und Wirkung sind, solange sie nur Korrelationen lernen. Er entwickelte den *do*-Kalkül und die Theorie der gerichteten azyklischen Graphen (DAGs), die es erlauben, kausale Fragen formal zu stellen und zu beantworten. Sein Buch "Causality: Models, Reasoning, and Inference" (2000) veränderte die Statistik, Epidemiologie, Oekonometrie und KI-Forschung. 2011 erhielt Pearl den Turing Award, den "Nobelpreis der Informatik".

Rauchen und Krebs: 50 Jahre Debatte
In den 1950er Jahren zeigten Richard Doll und Austin Bradford Hill eine starke Korrelation zwischen Rauchen und Lungenkrebs. Doch ausgerechnet Ronald Fisher – einer der größten Statistiker aller Zeiten – argumentierte dagegen: Vielleicht gebe es ein Gen, das sowohl die Lust aufs Rauchen als auch die Anfälligkeit für Krebs erhöhe. Die Tabakindustrie nutzte dieses Argument jahrzehntelang.
Das Problem: Ein randomisiertes Experiment war ethisch unmöglich – man konnte Menschen nicht zwingen zu rauchen. Erst als Bradford Hill 1965 seine neun Kriterien für Kausalität aus Beobachtungsdaten formulierte (Stärke, Konsistenz, Spezifität, Zeitlichkeit, Dosis-Wirkungs-Beziehung, Plausibilität, Kohärenz, Experiment, Analogie), konnte die wissenschaftliche Gemeinschaft den Zusammenhang als kausal anerkennen. Der US Surgeon General hatte bereits 1964 offiziell gewarnt. Es dauerte trotzdem Jahrzehnte, bis sich diese Erkenntnis gesellschaftlich durchsetzte.

Tyler Vigen: Margarine, Nicolas Cage und das Ertrinken
Tyler Vigen, damals Student an der Harvard Law School, startete 2014 das Projekt "Spurious Correlations". Er durchsuchte öffentliche Datensätze systematisch nach hohen Korrelationen zwischen völlig unzusammenhängenden Variablen. Seine Funde: Die Scheidungsrate im US-Bundesstaat Maine korreliert mit $r = 0{,}99$ mit dem Pro-Kopf-Verbrauch von Margarine. Die Anzahl der Filme mit Nicolas Cage korreliert mit der Zahl der Ertrinkungstode in Swimming Pools.
Vigens Projekt illustriert ein tiefes Problem: Wenn man genug Variablen und genug Zeitreihen hat, findet man *immer* hohe Korrelationen – rein durch Zufall. Das ist verwandt mit dem Problem des multiplen Testens (p-Hacking) aus Block 2: Bei 20 unabhängigen Tests mit $\alpha = 0{,}05$ ist ein "signifikantes" Ergebnis statistisch *erwartet*.

Häufige Fehler

Fehler 1: "Eine starke Korrelation bedeutet Kausalität."
Selbst $r = 0{,}99$ kann spurious sein. Tyler Vigens Margarine-Scheidungs-Korrelation ist $r = 0{,}99$, aber niemand würde behaupten, Margarine verursache Scheidungen. Die Stärke einer Korrelation sagt nichts über ihren kausalen Status aus. Entscheidend sind die fünf Kausalitätskriterien, insbesondere die Abwesenheit von Confounding und ein plausibler Mechanismus.

Fehler 2: "Wir können nie Kausalität zeigen, also brauchen wir es nicht zu versuchen."
Das ist zu pessimistisch. Randomisierte Experimente (A/B-Tests) können sehr wohl kausale Effekte nachweisen. Auch aus Beobachtungsdaten ist Kausalität unter bestimmten Annahmen schätzbar (Instrumentvariablen, DiD, Regression Discontinuity). Die richtige Haltung ist nicht "es geht nicht", sondern "welche Annahmen brauche ich, und wie plausibel sind sie?"

Fehler 3: "Kontrollvariablen lösen das Confounding-Problem."
Kontrollvariablen helfen nur gegen *bekannte und gemessene* Confounders. Unbekannte oder nicht gemessene Confounders können weiterhin verzerren. Deshalb ist die Annahme "no unmeasured confounding" so kritisch. Nur ein randomisiertes Experiment eliminiert alle Confounders – auch solche, an die niemand gedacht hat.

6.10 R-Code: Korrelation und partielle Korrelation

Korrelation und Signifikanztest

```r
# Korrelation zwischen Marketing und Retention
cor.test(dataco$marketing, dataco$retention)
# => r = 0.45, p = 0.027

# Korrelationsmatrix fuer mehrere Variablen
cor(dataco[, c("marketing", "retention", "konjunktur")])
```

Partielle Korrelation mit ppcor

```r
library(ppcor)

# Partielle Korrelation: Marketing und Retention,
# kontrolliert fuer Konjunktur
pcor_result <- pcor.test(
  x = dataco$marketing,
  y = dataco$retention,
  z = dataco$konjunktur
)

cat("Partielle Korrelation:", pcor_result$estimate)
# => 0.12
cat("p-Wert:",                pcor_result$p.value)
# => 0.38 (nicht signifikant!)
```

Einfacher DAG mit ggdag

```r
library(ggdag)

# DAG definieren
dag <- dagify(
  Retention ~ Marketing + Konjunktur,
  Marketing ~ Konjunktur,
  exposure = "Marketing",
  outcome  = "Retention"
)

# DAG zeichnen
ggdag(dag, text = TRUE, use_labels = "name") +
  theme_dag()

# Adjustment Set identifizieren
adjustmentSets(dag)
# => { Konjunktur }
```

6.11 Übungsaufgaben

Aufgabe 6.1 – Korrelation oder Kausalität?

Für jede der folgenden Beobachtungen: Handelt es sich wahrscheinlich um einen kausalen Zusammenhang oder eine Scheinkorrelation? Benennen Sie ggf. den Confounder.

(a) Schüler, die frühstücken, haben bessere Noten.

(b) Länder mit mehr Stöchen haben höhere Geburtenraten.

(c) Patienten, die Medikament X nehmen, haben niedrigeren Blutdruck.

(d) Städte mit mehr Feuerwehrleuten haben mehr Brände.

Aufgabe 6.2 – Pearls Leiter einordnen

Ordnen Sie folgende Aussagen den drei Stufen von Pearls Leiter (Sehen, Tun, Vorstellen) zu:

(a) “Kunden, die den Newsletter lesen, kaufen mehr.”

(b) “Wenn wir den Newsletter an alle schicken, steigt der Umsatz.”

(c) “Hätte Kunde X mehr gekauft, wenn er den Newsletter erhalten hätte?”

(d) “Raucher haben ein 20-fach höheres Lungenkrebsrisiko.”

Aufgabe 6.3 – DAG zeichnen

Zeichnen Sie einen DAG für folgende Situation: Ein Unternehmen beobachtet, dass Mitarbeiter, die Homeoffice machen, produktiver sind. Mögliche Confounders: Berufserfahrung (erfahrene Mitarbeiter dürfen eher ins Homeoffice) und Aufgabentyp (kreative Aufgaben werden öfter im Homeoffice erledigt).

(a) Welche Variablen sind Knoten?

(b) Welche Pfeile müssen Sie zeichnen?

(c) Was ist das Adjustment Set, um den kausalen Effekt von Homeoffice auf Produktivität zu schätzen?

Aufgabe 6.4 – Partielle Korrelation interpretieren

Lena beobachtet $r(\text{Werbung, Umsatz}) = 0{,}60$. Nach Kontrolle für "Saison" sinkt die partielle Korrelation auf $r_{\text{partiell}} = 0{,}55$. Nach zusätzlicher Kontrolle für "Wettbewerberaktivität" sinkt sie auf $r_{\text{partiell}} = 0{,}15$ (n.s.).

(a) Welcher Confounder hat den größeren Effekt: Saison oder Wettbewerber?

(b) Kann Lena jetzt schliessen, dass Werbung keinen kausalen Effekt auf den Umsatz hat?

(c) Welche Annahme müsste gelten, damit die partielle Korrelation den kausalen Effekt schätzt?

Aufgabe 6.5 – Experiment entwerfen

Lenas Chef möchte wissen, ob ein neues Treueprogramm die Kündigungsrate senkt.

(a) Warum reicht es nicht, die Kündigungsrate vor und nach der Einführung zu vergleichen?

(b) Entwerfen Sie ein randomisiertes Experiment. Definieren Sie Kontroll- und Treatmentgruppe, primäre Metrik und potenzielle Confounders.

(c) Welche Stufe auf Pearls Leiter erreichen Sie mit diesem Experiment?

Kernaussage: Ohne Experiment ist Kausalität schwer nachweisbar. Confounders können jede noch so starke Korrelation erzeugen. Bei Beobachtungsdaten immer nach Drittvariablen fragen, DAGs zeichnen, und im Zweifel einen A/B-Test planen.

7 Das Experiment – A/B-Testing

Lenas Herausforderung

Lena steht vor dem Board bei DataCo und präsentiert ihre Erkenntnis aus dem letzten Kapitel: Die Korrelation zwischen Marketing-Ausgaben und Kundenbindung war großteils durch die Konjunktur getrieben. Ihr Chef fragt: "Und jetzt? Wirkt unser Marketing oder nicht?" Lena antwortet: "Das können wir herausfinden – mit einem Experiment. Ich schlage einen A/B-Test vor: Wir teilen 1000 neue Kunden zufällig in zwei Gruppen. Gruppe A bekommt das alte Onboarding, Gruppe B das neue. Nach 30 Tagen vergleichen wir die Retention. Die Differenz ist dann der *kausale* Effekt des neuen Onboardings."

7.1 Entdeckungsfrage

Entdeckungsfrage

Lenas Chef runzelt die Stirn: "Warum brauchen wir 1000 Kunden für diesen Test? Reichen nicht 20?" Versuchen Sie, diese Frage zu beantworten, bevor Sie weiterlesen. Denken Sie an die Streuung in den Daten, an Zufall, und an das, was Sie über Konfidenzintervalle aus Block 2 wissen.

Die Antwort führt uns zu einem der zentralen Konzepte der experimentellen Forschung: der *statistischen Power*. Mit nur 20 Kunden wäre die Streuung so groß, dass selbst ein echter Effekt im Rauschen untergehen würde. Lena müsste genügend Beobachtungen sammeln, um den Effekt vom Zufall zu unterscheiden. Wie viele genau, das lässt sich vor dem Experiment berechnen – und genau das ist Lenas erster Schritt.

7.2 A/B-Test Prinzipien

A/B-Test: Randomisiertes Experiment mit zwei Gruppen (Kontrolle A und Treatment B), das kausale Schlussfolgerungen erlaubt

Ein A/B-Test folgt einem klaren Protokoll in vier Schritten:

1. **Zufällige Zuteilung:** Jeder Kunde wird per Zufallsmechanismus entweder Gruppe A oder Gruppe B zugewiesen. Keine Selbstselektion, keine Vorauswahl durch Manager.

2. **Intervention nur in Gruppe B:** Nur die Treatment-Gruppe erhält die Änderung – in Lenas Fall das neue Onboarding. Gruppe A durchläuft den bisherigen Prozess unverändert.

3. **Ergebnis messen:** Nach der festgelegten Laufzeit (hier 30 Tage) wird die primäre Metrik in beiden Gruppen gemessen.

4. **Differenz = kausaler Effekt:** $\Delta = \bar{Y}_B - \bar{Y}_A$ ist ein unverzerrter Schätzer des kausalen Effekts.

Warum funktioniert das? Die Antwort liegt in der Randomisierung.

Randomisierung: Zufällige Zuteilung zu Gruppen, die alle Confounders – auch unbekannte – im Erwartungswert balanciert

Wenn wir Kunden zufällig zuteilen, sind die beiden Gruppen im Erwartungswert identisch – nicht nur in den Variablen, die wir messen, sondern auch in allen Variablen, die wir nicht messen oder nicht einmal kennen. Das ist der entscheidende Unterschied zu Beobachtungsstudien: Die Randomisierung eliminiert *alle* Confounders, auch die unbekannten.

Kausaler Effekt

Der **Average Treatment Effect (ATE)** ist die mittlere Differenz im Outcome zwischen Treatment- und Kontrollgruppe unter Randomisierung:

$$\text{ATE} = E[\bar{Y}_B - \bar{Y}_A]$$

Unter Randomisierung ist diese Differenz ein unverzerrter Schätzer des kausalen Effekts, weil die Gruppen bis auf die Intervention identisch sind.

Stellen Sie sich zwei Kopien desselben Kundenuniversums vor: In einer Kopie erhält jeder das alte Onboarding, in der anderen das neue. Der kausale Effekt ist die Differenz zwischen diesen beiden kontrafaktischen Welten. Da wir nie beide Welten gleichzeitig beobachten können, nutzen wir Randomisierung als Nährung: Die Kontrollgruppe repräsentiert die Welt ohne Intervention, die Treatment-Gruppe die Welt mit Intervention.

7.3 Testdesign: Vor dem Experiment

Bevor Lena auch nur einen einzigen Kunden in den Test schickt, muss sie fünf Entscheidungen treffen:

1. **Hypothese formulieren:** "Das neue Onboarding erhöht die 30-Tage-Retention." Präzise und testbar, nicht vage.
2. **Primäre Metrik festlegen:** Retention nach 30 Tagen (binär: ja/nein). Nur *eine* primäre Metrik, nicht zehn.
3. **Minimum Detectable Effect (MDE) definieren:** Wie groß muss der Effekt sein, damit er geschäftlich relevant ist? Lena entscheidet: $d = 0{,}3$ Standardabweichungen – ein kleiner bis mittlerer Effekt.
4. **Stichprobengröße berechnen:** Basierend auf MDE, gewünschter Power und Signifikanzniveau.
5. **Laufzeit planen:** 30 Tage für die Intervention plus Puffer für Saisonalität.

MDE: Der kleinste Effekt, den der Test mit der geplanten Power entdecken kann.

Jede dieser Entscheidungen wird *vor* dem Test dokumentiert und nicht nachträglich angepasst. Dieser Grundsatz heißt *Pre-Registration* und schützt vor unbewusstem Anpassen der Analyse an die Ergebnisse.

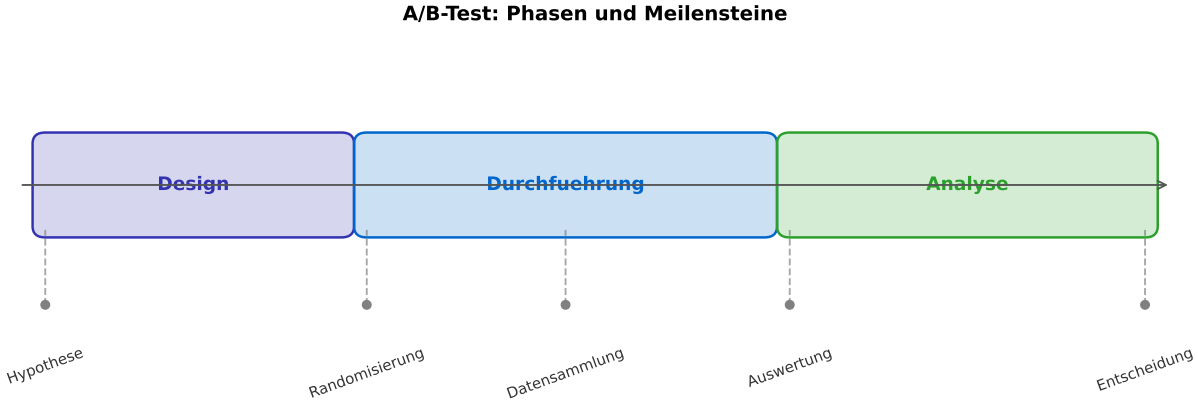


Abbildung 16: Zeitlicher Ablauf eines A/B-Tests: Design, Durchführung, Analyse

7.4 Stichprobengröße und Power

Stichprobengröße pro Gruppe

$$n \approx \frac{2\,(z_{\alpha/2} + z_{\beta})^2}{d^2}$$

Symbol für Symbol:

- n – benötigte Anzahl Beobachtungen *pro Gruppe* (nicht insgesamt)
- $z_{\alpha/2}$ – kritischer Wert der Standardnormalverteilung für das Signifikanzniveau (bei $\alpha = 0{,}05$ ist $z_{\alpha/2} = 1{,}96$)
- z_{β} – kritischer Wert für die gewünschte Power (bei Power $= 0{,}80$ ist $z_{\beta} = 0{,}84$; bei Power $= 0{,}90$ ist $z_{\beta} = 1{,}28$)
- d – die erwartete Effektstärke (Cohens d), also der minimale Effekt, den der Test entdecken soll

Die Formel gilt für einen zweiseitigen Zweistichproben-t-Test mit gleich großen Gruppen.

Power: Die Wahrscheinlichkeit, einen tatsächlich vorhandenen Effekt als statistisch signifikant zu erkennen. Standard: 80%

Lena setzt ihre Werte ein: $d = 0{,}3$, Power $= 0{,}80$ ($z_{\beta} = 0{,}84$), $\alpha = 0{,}05$ ($z_{\alpha/2} = 1{,}96$):

$$n \approx \frac{2 \times (1{,}96 + 0{,}84)^2}{0{,}3^2} = \frac{2 \times (2{,}80)^2}{0{,}09} = \frac{2 \times 7{,}84}{0{,}09} = \frac{15{,}68}{0{,}09} \approx 174 \text{ pro Gruppe}$$

Lena braucht also mindestens 174 Kunden in jeder Gruppe, um einen Effekt von $d = 0{,}3$ mit 80% Wahrscheinlichkeit zu entdecken. Warum plant sie dann 500 pro Gruppe? Drei Gründe: Erstens möchte sie Subgruppenanalysen durchführen (z. B. getrennt nach Premium- und Free-Kunden). Zweitens schützt ein größeres n gegen unvorhergesehene Datenverluste (Drop-outs, technische Fehler). Drittens steigt die Power nichtlinear – bei $n = 500$ beträgt sie bereits über 99% für $d = 0{,}3$.

Power für verschiedene Stichprobengrößen

Die folgende Tabelle zeigt, wie die Power mit wachsendem n steigt (bei $d = 0{,}3$, $\alpha = 0{,}05$):

n pro Gruppe	Power
20	0,12
50	0,25
100	0,48
174	0,80
300	0,95
500	0,99

Bei $n = 20$ würde Lena einen echten Effekt in weniger als jedem achten Versuch entdecken – sie würde in 88% der Fälle fälschlicherweise schlussfolgern, dass das neue Onboarding nicht wirkt.

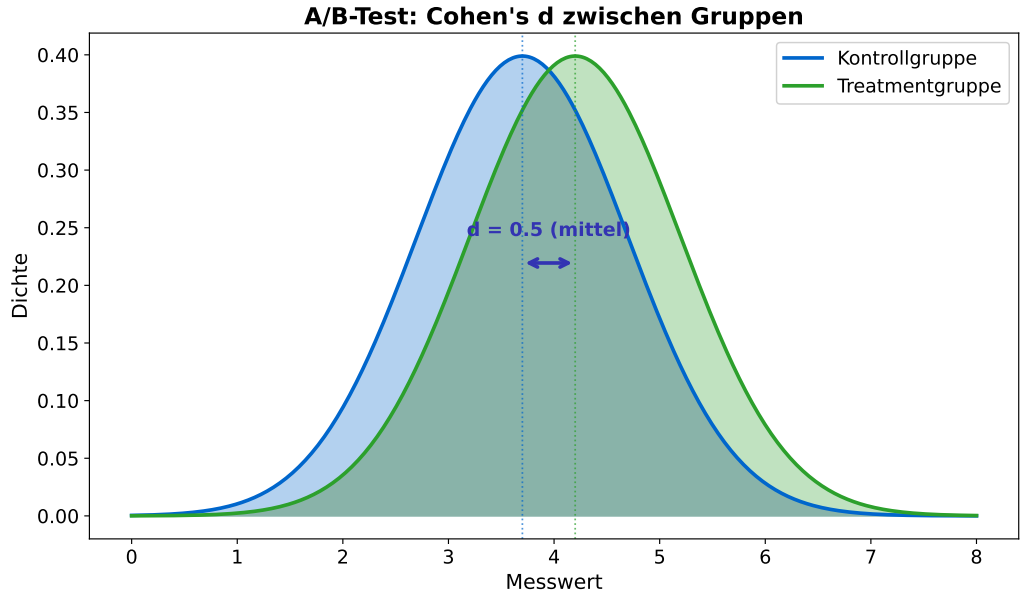


Abbildung 17: Verteilungen unter H_0 und H_1 mit Power und Signifikanzniveau

7.5 Cohens d: Effektstärke

Cohens d – standardisierte Effektstärke

$$d = \frac{\bar{X}_T - \bar{X}_C}{s_p}$$

Symbol für Symbol:

- d – die standardisierte Effektstärke (dimensionslos)
- $\bar{X}_T$ – der Mittelwert der Treatment-Gruppe
- $\bar{X}_C$ – der Mittelwert der Kontrollgruppe
- s_p – die gepoolte Standardabweichung, berechnet als:

$$s_p = \sqrt{\frac{(n_T - 1)\, s_T^2 + (n_C - 1)\, s_C^2}{n_T + n_C - 2}}$$

wobei s_T und s_C die Standardabweichungen in den beiden Gruppen sind

Cohens d: Standardisierte Effektstärke: Differenz der Mittelwerte geteilt durch die gepoolte Standardabweichung

Cohens d drückt den Unterschied zwischen zwei Gruppen in Einheiten der Standardabweichung aus. Das macht Effekte vergleichbar, selbst wenn die Originalskalen unterschiedlich sind (Minuten, Euro, Klicks). Jacob Cohen schlug 1988 folgende Daumenregel vor:

Cohens d	Interpretation
$d = 0{,}2$	kleiner Effekt – schwer mit bloßem Auge zu sehen
$d = 0{,}5$	mittlerer Effekt – deutlich spürbar
$d = 0{,}8$	großer Effekt – offensichtlich

Lenas späteres Ergebnis: $d = 0{,}45$. Das liegt zwischen klein und mittel und ist geschäftlich

relevant – fast eine halbe Standardabweichung Unterschied in der Retention. Zum Vergleich: Der Effekt einer Kopfschmerzaspirin auf Schmerzlinderung beträgt etwa $d = 0{,}5$.

Warum berichten wir nicht einfach den p-Wert? Weil der p-Wert von der Stichprobengröße abhängt: Mit einer Million Kunden wird selbst ein winziger, praktisch irrelevanter Unterschied "signifikant". Cohens d ist unabhängig von n und sagt uns, *wie groß* der Effekt ist, nicht nur, ob er existiert.

7.6 Drei gefährliche Fehler

Selbst ein sauber geplanter A/B-Test kann durch drei Fehler wertlos werden. Lena muss jeden einzelnen kennen.

Fehler 1: Peeking – zu früh reinschauen

Peeking: Wiederholtes Prüfen der Ergebnisse während eines laufenden Tests, was die Falsch-Positiv-Rate drastisch erhöht

Stellen Sie sich vor, Lena schaut nach einer Woche in die Daten: "Oh, $p = 0{,}03$! Signifikant! Wir können stoppen!" Das Problem: Wenn man bei jedem Zwischenblick testet, steigt die Wahrscheinlichkeit eines falsch-positiven Ergebnisses dramatisch an. Bei täglichem Prüfen über 30 Tage liegt die effektive Falsch-Positiv-Rate nicht bei 5%, sondern bei bis zu 30%.

Der Grund ist derselbe wie beim Münzwurf: Wenn Sie eine faire Münze oft genug werfen und jedes Mal prüfen, ob die bisherige Kopf-Quote signifikant von 50% abweicht, werden Sie irgendwann ein "signifikantes" Ergebnis finden – rein durch Zufall. Die Lösung: Stichprobengröße vorher festlegen und erst am Ende auswerten, oder spezielle sequentielle Testverfahren verwenden (z. B. Group Sequential Designs), die für Zwischenanalysen korrigieren.

Fehler 2: Multiple Testing – zu viele Metriken

Lenas Kollegin schlägt vor, neben der Retention auch Umsatz, Nutzungshäufigkeit, Support-Anfragen, App-Bewertungen, Klickrate, Seitenaufrufe, Session-Dauer, Empfehlungsrate und Newsletter-Abonnements zu messen. "Irgendwas wird schon signifikant!" Das ist genau das Problem.

Bei 10 unabhängigen Tests auf dem 5%-Niveau beträgt die Wahrscheinlichkeit, *mindestens einen* falsch-positiven Befund zu erhalten:

$$1 - (1 - 0{,}05)^{10} = 1 - 0{,}95^{10} \approx 0{,}40$$

Eine 40%-ige Chance auf ein falsch-positives Ergebnis! Die Lösung: *Eine* primäre Metrik vorher festlegen. Weitere Metriken dürfen explorativ betrachtet werden, aber nur mit Korrektur für Mehrfachtests (z. B. Bonferroni: α/k für k Tests).

Fehler 3: Simpson's Paradoxon

Simpson's Paradoxon: Ein Effekt, der in Subgruppen in die entgegengesetzte Richtung zeigt wie in den Gesamtdaten

Lenas A/B-Test zeigt insgesamt einen positiven Effekt des neuen Onboardings. Aber als sie nach Premium- und Free-Kunden aufschlüsselt, entdeckt sie Überraschendes: In *beiden* Subgruppen einzeln ist der Effekt negativ! Wie kann das sein?

Das Paradoxon entsteht durch unterschiedliche Gruppenzusammensetzung. Wenn in der Treatment-Gruppe zufällig mehr Premium-Kunden gelandet sind (die ohnehin höhere Retention haben), kann der Gesamteffekt positiv aussehen, obwohl das neue Onboarding in jeder Subgruppe schadet. Dieses Phänomen heißt Simpson's Paradoxon und ist der Grund, warum Lena nach dem Test immer Subgruppenanalysen durchführt und die Randomisierung auf Balance prüft.

7.7 Beispiel: Lenas vollständiger A/B-Test

Lenas DataCo A/B-Test: Vom Design zur Board-Präsentation

Phase 1 – Design: Lena legt fest: H_0: "Das neue Onboarding hat keinen Effekt auf die 30-Tage-Retention." Primäre Metrik: Retention (binär). MDE: $d = 0{,}3$. Power: 80%. $\alpha = 0{,}05$. Benötigte Stichprobengröße: $n = 174$ pro Gruppe, geplant: 500 pro Gruppe.
Phase 2 – Durchführung: 1000 neue Kunden werden per Zufallsgenerator in Gruppe A (altes Onboarding) und Gruppe B (neues Onboarding) eingeteilt. Lena prüft die Randomisierung: Alter, Geschlecht, Premium-Status sind in beiden Gruppen fast identisch verteilt. Während der 30 Tage schaut Lena *nicht* in die Daten.
Phase 3 – Auswertung:

	Kontrolle (A)	**Treatment (B)**
30-Tage-Retention	68%	76%
Mittlerer Zufriedenheits-Score	3,7	4,2

Ergebnisse: $d = 0{,}45$ (mittlerer Effekt), $p = 0{,}003$, 95%-Konfidenzintervall für d: [0,18; 0,72].
Phase 4 – Interpretation: Das Konfidenzintervall schliesst Null nicht ein, der Effekt ist statistisch signifikant *und* praktisch relevant. Das neue Onboarding erhöht die 30-Tage-Retention um 8 Prozentpunkte (von 68% auf 76%), was einer standardisierten Effektstärke von $d = 0{,}45$ entspricht.
Phase 5 – Board-Präsentation: Lena empfiehlt dem Board, das neue Onboarding für alle Kunden auszurollen. Bei 10.000 Neukunden pro Monat und einem Customer Lifetime Value von 500 CHF entsprechen die 8 Prozentpunkte mehr Retention einem geschätzten Mehrwert von 400.000 CHF pro Monat.

Von Linds Skorbut-Versuch (1747) zu Googles 10.000 Tests pro Jahr

Die Geschichte des kontrollierten Experiments beginnt auf hoher See. 1747 teilte der schottische Schiffsarzt James Lind zwölf Skorbut-Patienten auf der HMS Salisbury in sechs Gruppen ein und gab jeder eine andere Behandlung: Zitrusfrüchte, Essig, Meerwasser, Apfelwein, Schwefelsäure und eine Paste aus Knoblauch und Senf. Nur die Gruppe mit Zitrusfrüchten erholte sich. Obwohl Lind noch nichts von Randomisierung wusste, hatte er das Grundprinzip geschaffen: gleiche Bedingungen bis auf die Intervention, dann vergleichen.
Fast 200 Jahre später formalisierte Ronald Fisher in den 1920er-Jahren die Idee der Randomisierung in seiner Arbeit über Agrarexperimente. Seine berühmte "Lady Tasting Tea"-Geschichte – in der eine Dame behauptete, schmecken zu können, ob die Milch vor oder nach dem Tee in die Tasse gegossen wurde – führte zur Entwicklung des exakten Tests und der modernen Versuchsplanung.
Im digitalen Zeitalter explodierte die Anwendung: Barack Obamas Wahlkampfteam testete 2008 insgesamt 24 Versionen seiner Spendenseite per A/B-Test. Die Gewinnerversion brachte 60 Millionen Dollar mehr Spenden als die Originalversion ein. Heute führt Google über 10.000 A/B-Tests pro Jahr durch, Amazon, Netflix und Booking.com testen praktisch jede Änderung an ihren Plattformen.

Häufige Fehler

Irrtum 1: "A/B-Tests geben immer klare Antworten." Nein. Bei kleinen Effekten und zu kleinen Stichproben liefern A/B-Tests oft nicht-signifikante Ergebnisse – das bedeutet nicht, dass kein Effekt existiert, sondern dass der Test zu wenig Power hatte. Ein nicht-signifikantes Ergebnis ist kein Beweis für die Abwesenheit eines Effekts.
Irrtum 2: "Statistisch signifikant = praktisch wichtig." Nein. Bei sehr großen Stichproben wird selbst ein winziger Unterschied von 0,1 Prozentpunkt signifikant. Deshalb immer die Effektstärke (Cohens d) berichten, nicht nur den p-Wert. Ein Ergebnis mit $p = 0{,}001$ aber $d = 0{,}02$ ist statistisch signifikant, aber geschäftlich irrelevant.
Irrtum 3: "Früh reinschauen ist in Ordnung, solange man nicht stoppt." Falsch. Allein das Anschauen der Daten und die mentale Aktualisierung der Überzeugung beeinflussen die Entscheidungsfindung. Außerdem erhöht jedes Prüfen die kumulative Fehlerrate. Die Regel lautet: Entweder am Ende auswerten oder ein sequentielles Testverfahren verwenden, das für Zwischenanalysen korrigiert.

7.8 R-Code: t.test, Cohens d, und Power-Analyse

R-Code

```r
# --- A/B-Test Auswertung ---

# Daten laden (Lenas DataCo Experiment)
library(tidyverse)

ab_daten <- tibble(
  gruppe = rep(c("Kontrolle", "Treatment"), each = 500),
  retention = c(
    rbinom(500, 1, 0.68),   # Kontrolle: 68%
    rbinom(500, 1, 0.76)    # Treatment: 76%
  )
)

# Deskriptive Statistik
ab_daten %>%
  group_by(gruppe) %>%
  summarise(
    n = n(),
    retention_rate = mean(retention),
    se = sd(retention) / sqrt(n())
  )

# Zweistichproben-t-Test
t.test(retention ~ gruppe, data = ab_daten)

# Cohen's d mit dem effsize-Paket
library(effsize)
cohen.d(retention ~ gruppe, data = ab_daten)

# Power-Analyse: Wie viele Kunden brauchen wir?
power.t.test(
  delta = 0.3,      # Erwarteter Cohen's d
  sd = 1,           # Standardisiert
  sig.level = 0.05,
  power = 0.80,
  type = "two.sample"
)
# Ergebnis: n = 176 pro Gruppe (numerisch, leicht ueber 174)

# Manuelle Berechnung der Sample Size
z_alpha <- qnorm(1 - 0.05/2)  # 1.96
z_beta  <- qnorm(0.80)        # 0.84
d <- 0.3
n_manual <- ceiling(2 * ((z_alpha + z_beta) / d)^2)
cat("Benoetigte n pro Gruppe:", n_manual, "\n")
```

7.9 Übungsaufgaben

Aufgabe 7.1 – Power-Analyse

Ein Online-Shop möchte testen, ob eine neue Produktseite die Conversion Rate erhöht. Der erwartete Effekt ist $d = 0{,}4$. Wie viele Besucher braucht man pro Gruppe bei $\alpha = 0{,}05$ und Power $= 0{,}80$? Setzen Sie die Werte in die Formel ein und rechnen Sie Schritt für Schritt.

Aufgabe 7.2 – Cohens d berechnen

In einem A/B-Test beträgt der mittlere Umsatz in der Kontrollgruppe $\bar{X}_C = 48$ CHF mit $s_C = 12$ CHF ($n_C = 200$) und in der Treatment-Gruppe $\bar{X}_T = 52$ CHF mit $s_T = 14$ CHF ($n_T = 200$). Berechnen Sie die gepoolte Standardabweichung und Cohens d. Wie ordnen Sie das Ergebnis nach Cohens Daumenregel ein?

Aufgabe 7.3 – Peeking-Problem

Ein Produktmanager schaut während eines A/B-Tests jeden Tag in die Daten und stoppt den Test nach 8 Tagen, weil $p = 0{,}04$. Der Test war für 30 Tage geplant. Erklären Sie, warum dieses Ergebnis nicht verlässlich ist, und berechnen Sie die ungefähre effektive Falsch-Positiv-Rate bei 8 Zwischenanalysen (Hinweis: verwenden Sie $1 - 0{,}95^k$ als Näherung).

Aufgabe 7.4 – Multiple Testing

Ein Team testet in einem A/B-Test gleichzeitig 15 verschiedene Metriken auf dem 5%-Niveau. Wie hoch ist die Wahrscheinlichkeit, mindestens ein falsch-positives Ergebnis zu finden? Welches Signifikanzniveau würde die Bonferroni-Korrektur für jede einzelne Metrik vorgeben?

Aufgabe 7.5 – Vollständiger Test

Sie planen einen A/B-Test für eine neue E-Mail-Kampagne. Der erwartete Effekt auf die Klickrate ist $d = 0{,}25$. Beschreiben Sie alle fünf Designschritte (Hypothese, Metrik, MDE, Sample Size, Laufzeit) und begründen Sie Ihre Entscheidungen.

Kernaussage: A/B-Tests sind der Goldstandard für kausale Fragen im Business. Der Schlüssel liegt im Design: Hypothese, primäre Metrik und Effektstärke *vor* dem Test festlegen. Die drei größten Fallen sind Peeking, Multiple Testing und Simpson's Paradoxon.

8 Ja oder Nein – Logistische Regression

Lenas Herausforderung

Lena hat in den letzten Wochen bei DataCo viel gelernt: Multikollinearität erkennen, Modelle regularisieren, mit Kreuzvalidierung validieren, und mit einem A/B-Test einen kausalen Effekt nachweisen. Jetzt steht sie vor einer neuen Herausforderung: Churn. Der Kunde kündigt – oder er kündigt nicht. Ja oder Nein. Als Lena versucht, eine lineare Regression auf diese binäre Zielvariable anzuwenden, passiert etwas Merkwuerdiges: Für manche Kunden sagt das Modell eine Kündigungswahrscheinlichkeit von $-0{,}12$ vorher, für andere 1,35. Negative Wahrscheinlichkeiten? Wahrscheinlichkeiten über 100%? Das kann nicht stimmen. Lena braucht ein anderes Modell.

8.1 Entdeckungsfrage

Entdeckungsfrage

Warum versagt OLS bei Ja/Nein-Fragen? Zeichnen Sie sich eine Gerade durch Datenpunkte, die nur bei $Y = 0$ und $Y = 1$ liegen. Was passiert an den Rändern?

8.2 Lichtschalter vs. Dimmer

Die Analogie hilft, das Grundproblem zu verstehen. Churn ist wie ein Lichtschalter

Binäre Variable: Variable, die nur zwei Werte annimmt (0/1, Ja/Nein, Churn/Kein Churn)

– der Kunde geht oder bleibt, es gibt kein Dazwischen. Aber wir wollen etwas Subtileres modellieren: die *Wahrscheinlichkeit*, dass der Schalter auf "Kündigung" umgelegt wird. Diese Wahrscheinlichkeit ist kontinuierlich und liegt zwischen 0 und 1 – wie ein Dimmer. Je weiter der Dimmer aufgedreht ist, desto heller das Licht, desto wahrscheinlicher die Kündigung.

Das Ziel der logistischen Regression ist genau dieser Übergang: Wir modellieren nicht die binäre Entscheidung selbst, sondern die Wahrscheinlichkeit $P(Y = 1|X)$ – also die Wahrscheinlichkeit der Kündigung, gegeben die Merkmale des Kunden. Diese Wahrscheinlichkeit muss immer zwischen 0 und 1 liegen.

8.3 Warum OLS bei binären Daten versagt

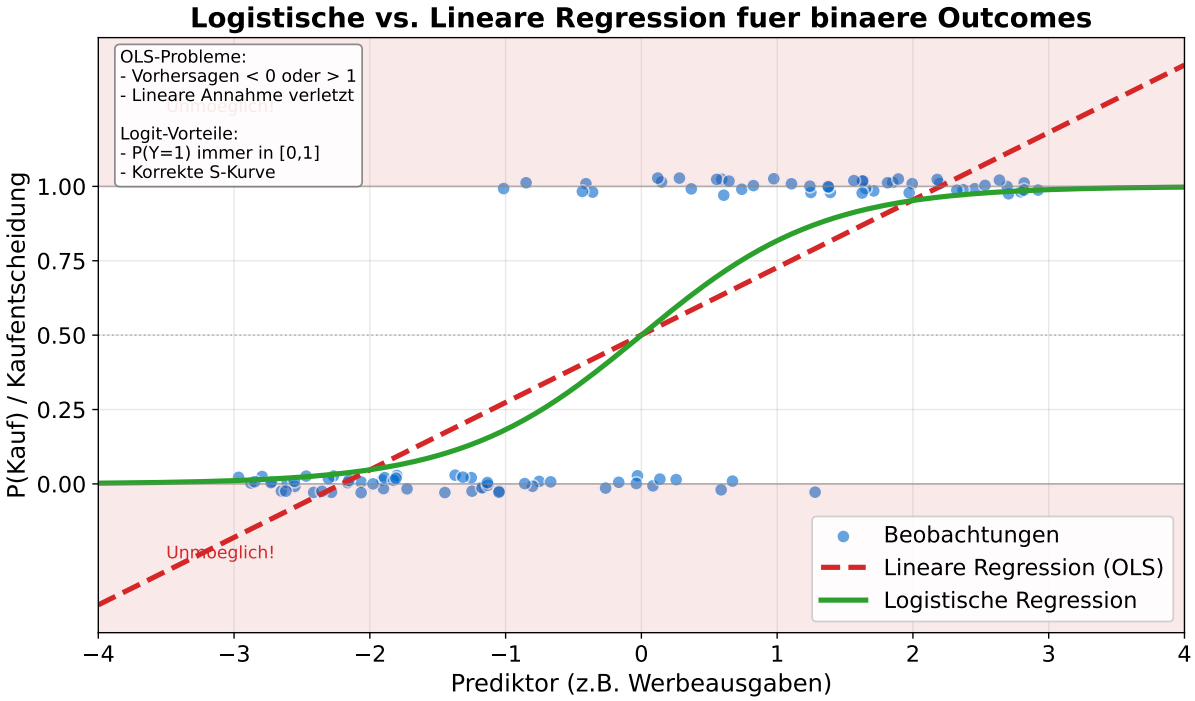
Lineares Wahrscheinlichkeitsmodell: OLS-Regression auf binäre Zielvariable – einfach, aber mit drei fundamentalen Problemen

Wenn wir eine gewöhnliche lineare Regression $P(Y = 1|X) = \beta_0 + \beta_1 X$ auf binäre Daten anwenden, treten drei fundamentale Probleme auf:

Erstens liegen die Vorhersagen nicht zwingend im Intervall $[0, 1]$. Eine Gerade ist unbeschränkt – für extreme X-Werte sagt das Modell negative Wahrscheinlichkeiten oder Wahrscheinlichkeiten über 1 vorher. Bei Lenas Daten ergibt sich für Kunden mit sehr hoher Nutzungsdauer eine vorhergesagte Kündigungswahrscheinlichkeit von $-0{,}12$: mathematischer Unsinn.

Zweitens ist die Linearitätsannahme verletzt. Die wahre Beziehung zwischen X und $P(Y = 1)$ ist in der Regel nicht linear, sondern S-förmig: Bei sehr niedrigen und sehr hohen X-Werten ändert sich die Wahrscheinlichkeit kaum, in der Mitte ändert sie sich stark.

Drittens verletzt das Modell die Annahme der Homoskedastizität. Wenn Y binär ist, hängt die Varianz der Residuen vom Mittelwert ab: $\text{Var}(Y|X) = P(1 - P)$, was bei $P = 0{,}5$ maximal ist und gegen 0 und 1 abnimmt.



Die Abbildung zeigt den Unterschied: Die rote Gerade (OLS) überschreitet die Grenzen, die blaue S-Kurve (logistische Regression) bleibt immer im Intervall $[0, 1]$.

8.4 Die logistische Funktion

Die Lösung ist elegant: Statt P direkt als lineare Funktion von X zu modellieren, transformieren wir die lineare Kombination $\beta_0 + \beta_1 X$ durch die logistische Funktion (auch Sigmoid-Funktion genannt):

Die logistische Funktion

$$P(Y = 1|X) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X)}}$$

Symbol für Symbol:

- $P(Y = 1|X)$ – die Wahrscheinlichkeit, dass $Y = 1$ (z. B. Kündigung), gegeben den Prädiktor X
- e – die Eulersche Zahl ($\approx 2{,}718$), Basis des natürlichen Logarithmus
- β_0 – der Intercept (Achsenabschnitt auf der Log-Odds-Skala)
- β_1 – der Koeffizient von X (Änderung der Log-Odds pro Einheit X)
- $\beta_0 + \beta_1 X$ – der lineare Prädiktor, der jeden Wert von $-\infty$ bis $+\infty$ annehmen kann

Sigmoid-Funktion: S-förmige Funktion, die jeden reellen Wert in das Intervall $(0, 1)$ abbildet

Die S-Kurve hat drei wichtige Eigenschaften. Erstens ist sie auf das Intervall $(0, 1)$ beschränkt – egal wie extrem die X-Werte sind, die vorhergesagte Wahrscheinlichkeit liegt immer zwischen 0 und 1. Zweitens ist sie symmetrisch um den Wendepunkt bei $P = 0{,}5$. Drittens ist die Kurve am steilsten bei $P = 0{,}5$ und flacht zu den Rändern hin ab – kleine Änderungen in X haben den größten Effekt auf P, wenn die Wahrscheinlichkeit um 50% liegt, und kaum Effekt, wenn sie bereits nahe 0 oder 1 ist.

Anschaulich: Wenn ein Kunde bereits eine Kündigungswahrscheinlichkeit von 95% hat, ändert ein zusätzlicher Beschwerdeanruf kaum noch etwas. Wenn die Wahrscheinlichkeit bei 50% liegt, kann derselbe Anruf den Ausschlag geben.

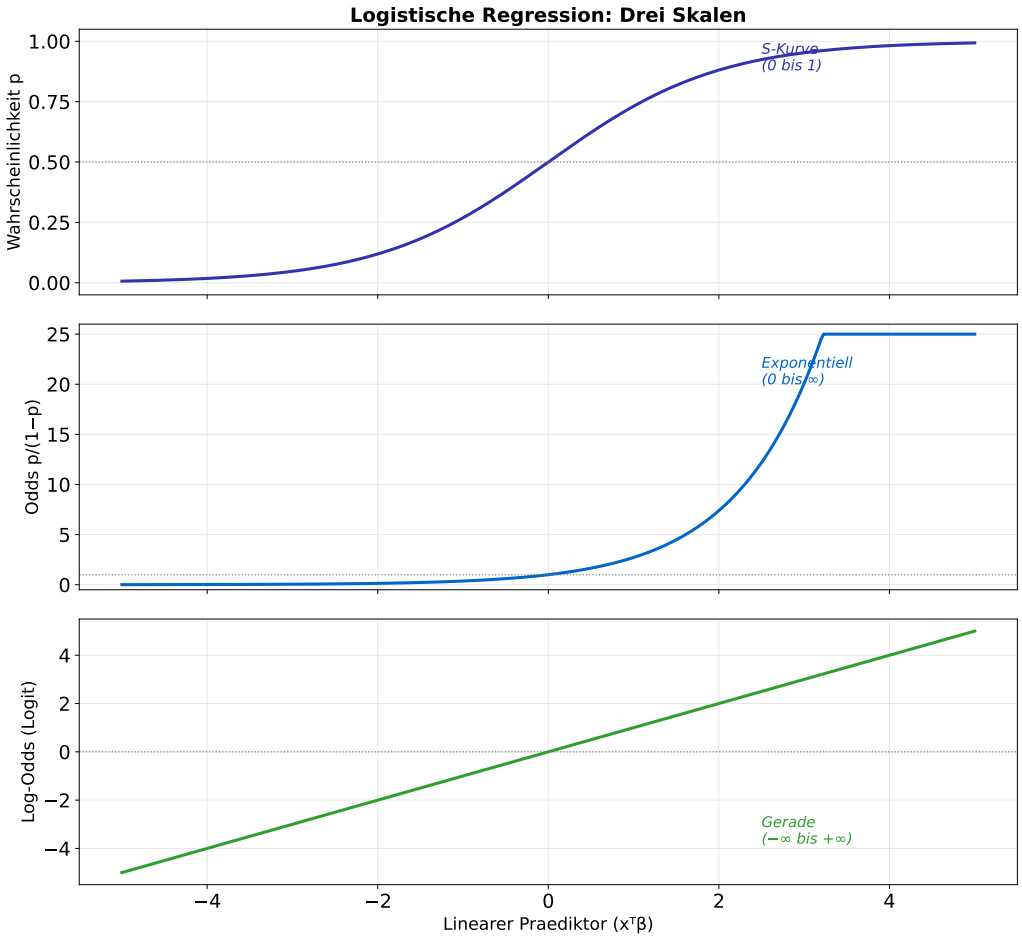


Abbildung 18: Drei Skalen der logistischen Regression: Wahrscheinlichkeit, Odds, Log-Odds

8.5 Log-Odds und Odds Ratios

Die logistische Funktion lässt sich umkehren, und die Umkehrung führt zur Logit-Transformation, die das Modell interpretierbar macht:

Logit-Transformation und Odds Ratio

Log-Odds (Logit):

$$\ln \frac{p}{1-p} = \beta_0 + \beta_1 X$$

Odds Ratio:

$$OR = e^{\beta_1}$$

Symbol für Symbol:

- p – Kurzschreibweise für $P(Y = 1|X)$, die Wahrscheinlichkeit des Ereignisses
- $\frac{p}{1-p}$ – die **Odds** ("Chancenverhältnis"): das Verhältnis der Wahrscheinlichkeit für das Ereignis zur Wahrscheinlichkeit dagegen. Bei $p = 0{,}80$ sind die Odds $\frac{0{,}80}{0{,}20} = 4$ ("4 zu 1")
- $\ln \frac{p}{1-p}$ – die **Log-Odds** (Logit): der natürliche Logarithmus der Odds. Dieser Wert ist unbeschränkt ($-\infty$ bis $+\infty$) und linear in X
- β_1 – die Änderung der Log-Odds pro Einheit X
- e^{β_1} – die **Odds Ratio**: der Faktor, um den sich die Odds multiplizieren, wenn X um eine Einheit steigt

Odds: Verhältnis $p/(1-p)$, z. B. Odds = 4 bedeutet "4 zu 1 für das Ereignis"

Odds Ratio (OR): e^{β_1}: Faktor, um den sich die Odds bei einer Einheit mehr X verändern

Das Konzept der Odds mag zunächst ungewohnt wirken, ist aber in vielen Lebensbereichen gebräuchlich. Bei Sportwetten spricht man von "3 zu 1 gegen" – das sind Odds von $1/3 = 0{,}33$. Bei Wahrscheinlichkeit $p = 0{,}75$ sind die Odds $0{,}75/0{,}25 = 3$: "Dreimal so wahrscheinlich wie unwahrscheinlich."

Der Clou der logistischen Regression ist: Auf der Log-Odds-Skala ist das Modell *linear*. Die Koeffizienten β beschreiben die Änderung der Log-Odds. Und die Exponentialfunktion e^{β} übersetzt diese Änderung in einen multiplikativen Faktor für die Odds – das ist die Odds Ratio.

Lenas Premium-Kunden: Von Log-Odds zur Wahrscheinlichkeit

In Lenas DataCo-Modell beträgt der Koeffizient für "Premium-Abo" $\beta_{\text{premium}} = 0{,}916$.
Schritt 1 – Odds Ratio berechnen: $OR = e^{0{,}916} = 2{,}5$
Premium-Kunden haben 2,5-fach höhere Odds zu bleiben als Non-Premium-Kunden.
Schritt 2 – Vom OR zur Wahrscheinlichkeit: Nehmen wir an, die Basis-Bleibewahrscheinlichkeit (Non-Premium) ist $p = 0{,}40$.
Die Basis-Odds sind: $\frac{0{,}40}{0{,}60} = 0{,}667$
Die Premium-Odds sind: $0{,}667 \times 2{,}5 = 1{,}667$
Die Premium-Wahrscheinlichkeit ist: $p = \frac{1{,}667}{1+1{,}667} = \frac{1{,}667}{2{,}667} \approx 0{,}63$
Fazit: Das Premium-Abo erhöht die Bleibewahrscheinlichkeit von 40% auf 63%.

8.6 Confusion Matrix

Confusion Matrix: Tabelle, die die vorhergesagten Klassen mit den tatsächlichen Klassen vergleicht (TP, FP, TN, FN)

Wenn wir die vorhergesagten Wahrscheinlichkeiten in binäre Entscheidungen übersetzen (z. B. "Churn" wenn $P > 0{,}5$), entsteht eine 2×2-Tabelle, die sogenannte Confusion Matrix:

	Vorhergesagt: Bleibt	**Vorhergesagt: Kündigt**
Tatsächlich: Bleibt	TN (True Negative)	FP (False Positive)
Tatsächlich: Kündigt	FN (False Negative)	TP (True Positive)

In Lenas Kontext bei DataCo (15% der Kunden kündigen):

- **TP (True Positive):** Das Modell sagt "Kündigt" und der Kunde kündigt tatsächlich. Lena kann rechtzeitig intervenieren.
- **FP (False Positive):** Das Modell sagt "Kündigt", aber der Kunde bleibt. Lena verschwendet Ressourcen auf zufriedene Kunden.
- **TN (True Negative):** Das Modell sagt "Bleibt" und der Kunde bleibt. Kein Handlungsbedarf.
- **FN (False Negative):** Das Modell sagt "Bleibt", aber der Kunde kündigt. Das ist der teuerste Fehler – Lena übersieht gefährdete Kunden.

Aus der Confusion Matrix lassen sich vier zentrale Metriken ableiten:

Klassifikationsmetriken

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Anteil aller korrekt klassifizierten Fälle.

$$\text{Precision} = \frac{TP}{TP + FP}$$

Von allen als "Kündiger" markierten: wie viele sind es tatsächlich?

$$\text{Recall (Sensitivity)} = \frac{TP}{TP + FN}$$

Von allen tatsächlichen Kündigern: wie viele hat das Modell erkannt?

$$F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Harmonisches Mittel von Precision und Recall – ein Kompromiss.

Lenas DataCo Confusion Matrix

Lenas Modell auf $n = 500$ Testkunden (davon 75 tatsächliche Kündiger):

	Pred. Bleibt	**Pred. Kündigt**	**Summe**
Bleibt	400 (TN)	25 (FP)	425
Kündigt	19 (FN)	56 (TP)	75
Summe	419	81	500

Metriken:

- Accuracy $= \frac{56+400}{500} = 0{,}912$ (91,2%)
- Precision $= \frac{56}{56+25} = 0{,}691$ (69,1%)
- Recall $= \frac{56}{56+19} = 0{,}747$ (74,7%)
- $F1 = 2 \cdot \frac{0{,}691 \cdot 0{,}747}{0{,}691+0{,}747} = 0{,}718$ (71,8%)

Beachten Sie: Die Accuracy von 91,2% klingt beeindruckend, aber ein triviales Modell, das einfach *immer* "Bleibt" vorhersagt, würde bereits 425/500 = 85% Accuracy erreichen! Bei unbalancierten Klassen (hier 15% Kündiger) ist Accuracy irreführend. Precision, Recall und $F1$ sind aussagekräftiger.

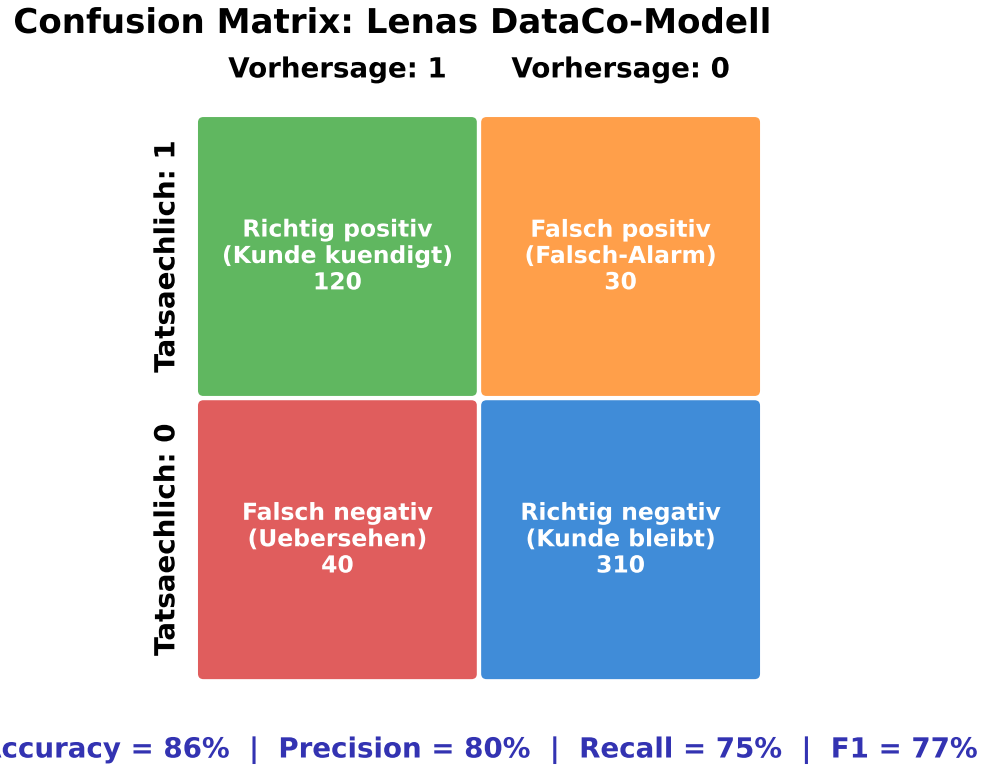


Abbildung 19: Visuelle Darstellung der Confusion Matrix mit Lenas Churn-Daten

8.7 ROC-Kurve und AUC

ROC-Kurve: Receiver Operating Characteristic: zeigt für jeden Schwellenwert die True Positive Rate vs. die False Positive Rate

Bisher haben wir den Schwellenwert $P > 0{,}5$ verwendet, um Kunden als "Kündiger" zu klassifizieren. Aber warum genau 0,5? Wenn wir den Schwellenwert senken (z. B. auf 0,3), erkennen wir mehr tatsächliche Kündiger (höhere Recall), produzieren aber auch mehr Fehlalarme (niedrigere Precision). Wenn wir den Schwellenwert erhöhen (z. B. auf 0,7), machen wir weniger Fehlalarme, übersehen aber mehr Kündiger.

Die ROC-Kurve visualisiert diesen Tradeoff: Sie zeichnet für *jeden* möglichen Schwellenwert die True Positive Rate (Recall) gegen die False Positive Rate ($= FP/(FP+TN)$). Ein perfektes Modell hätte eine ROC-Kurve, die durch die obere linke Ecke geht (100% Recall bei 0% False Positives). Ein zufälliges Modell läge auf der Diagonalen.

AUC: Area Under the ROC Curve: Wahrscheinlichkeit, dass das Modell einen zufälligen Positiven höher rankt als einen zufälligen Negativen

Die AUC (Area Under the Curve) fasst die ROC-Kurve in einer einzigen Zahl zusammen. Die Interpretation ist elegant: Die AUC ist die Wahrscheinlichkeit, dass das Modell einem zufällig ausgewählten tatsächlichen Kündiger eine höhere Kündigungswahrscheinlichkeit zuweist als einem zufällig ausgewählten Bleiber. Bei Lenas Modell beträgt die AUC = 0,82: In 82% der Fälle rankt ihr Modell einen Kündiger höher als einen Bleiber.

AUC	**Interpretation**
0,5	Zufall – das Modell unterscheidet nicht besser als Münzwurf
0,7–0,8	Akzeptabel – brauchbar für erste Screening-Zwecke
0,8–0,9	Gut – zuverlässig für die meisten Business-Anwendungen
$> 0{,}9$	Exzellent – hohe Trennschärfe, typisch für medizinische Diagnostik

Lenas AUC von 0,82 liegt im "gut"-Bereich. Für ein erstes Churn-Modell mit wenigen Features ist das ein solides Ergebnis. Verbesserungen wären möglich durch zusätzliche Features (z. B. Nutzungsverhalten, Support-Interaktionen) oder komplexere Modelle (Random Forest, Gradient Boosting – das lernt Lena im nächsten Modul).

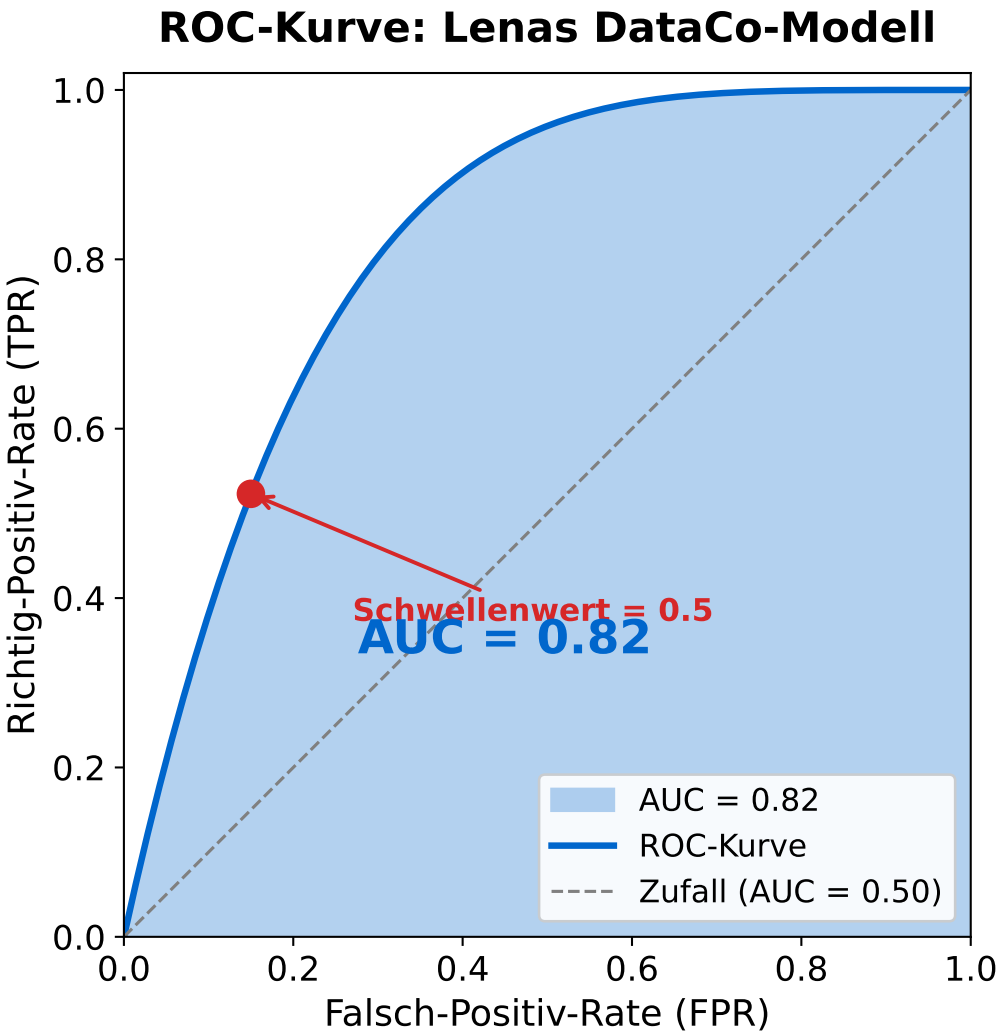


Abbildung 20: ROC-Kurve mit AUC für Lenas Churn-Modell

8.8 Beispiele: DataCo und Titanic

Lenas DataCo Logistic Regression: Vollständige Interpretation

Lena schätzt folgendes Modell:

```
glm(churn ~ premium + alter + nutzung_monate + beschwerden,
                data = dataco, family = binomial)
```

Output (vereinfacht):

Variable	Koeff. (β)	OR (e^{β})	95%-KI	p-Wert
(Intercept)	−2,00	–	–	$< 0{,}001$
Premium	−0,92	0,40	[0,27; 0,59]	$< 0{,}001$
Alter (pro Jahr)	0,02	1,02	[1,01; 1,03]	0,018
Nutzung (Monate)	−0,05	0,95	[0,93; 0,97]	0,002
Beschwerden	0,30	1,35	[1,18; 1,54]	$< 0{,}001$

Interpretation Zeile für Zeile:

- **Premium** ($OR = 0{,}40$): Premium-Kunden haben nur 40% der Kündigungs-Odds von Non-Premium-Kunden – das Premium-Abo ist der stärkste Schutzfaktor.
- **Alter** ($OR = 1{,}02$): Pro Lebensjahr steigen die Kündigungs-Odds um 2%. Ein kleiner Effekt, aber über große Altersspannen kumulativ relevant.
- **Nutzung** ($OR = 0{,}95$): Pro zusätzlichem Monat Nutzung sinken die Kündigungs-Odds um 5%. Langjährige Kunden kündigen seltener.
- **Beschwerden** ($OR = 1{,}35$): Jede zusätzliche Beschwerde erhöht die Kündigungs-Odds um 35%. Drei Beschwerden: $1{,}35^3 = 2{,}46$-fache Odds.

Lenas Empfehlung an das Management: Premium-Kunden binden und das Beschwerde-Management verbessern – das sind die größten Hebel.

Titanic: Das "Hello World" der Logistischen Regression

Die Titanic sank 1912 mit 2224 Passagieren, von denen 710 überlebten. Der Titanic-Datensatz ist das vielleicht bekannteste Lehrbeispiel für logistische Regression und zeigt, wie gesellschaftliche Strukturen in Daten sichtbar werden.
Eine logistische Regression auf die Überlebensvariable ergibt:

Feature	OR	Interpretation
Geschlecht (weiblich)	10,5	Frauen hatten 10-fach höhere Überlebens-Odds als Männer
1. Klasse (vs. 3.)	3,8	Passagiere der 1. Klasse überlebten deutlich öfter
Alter (pro Jahr)	0,97	Aeltere waren leicht benachteiligt
Kinder (< 10 J.)	2,1	"Frauen und Kinder zuerst" – in den Daten nachweisbar

Die Parallele zu DataCo ist klar: Was bei der Titanic Klasse und Geschlecht sind, sind bei Lena Premium-Abo und Nutzungsintensität. Die logistische Regression deckt in beiden Fällen auf, welche Faktoren den Ausgang beeinflussen – und wie stark.

8.9 Anwendungsgebiete

Die logistische Regression gehört zu den am häufigsten eingesetzten Modellen in der angewandten Data Science. Vier zentrale Anwendungsfelder:

Churn Prediction – Lenas Anwendung bei DataCo. Welche Kunden werden in den nächsten 30 Tagen kündigen? Die vorhergesagte Wahrscheinlichkeit ermöglicht gezielte Retention-Maßnahmen für die gefährdetsten Kunden.

Conversion Prediction – Welcher Website-Besucher wird kaufen? Online-Shops verwenden logistische Regression, um in Echtzeit zu entscheiden, ob ein Besucher einen Rabatt-Banner sehen soll (bei hoher Kaufwahrscheinlichkeit nicht nötig, bei mittlerer den Ausschlag gebend).

Credit Scoring – Banken bewerten seit Jahrzehnten die Ausfallwahrscheinlichkeit von Krediten mit logistischer Regression. Die Interpretierbarkeit durch Odds Ratios ist hier regulatorisch vorgeschrieben (Basel-Richtlinien): Die Bank muss erklären können, *warum* ein Kredit abgelehnt wurde.

Fraud Detection – Ist eine Kreditkartentransaktion betrügerisch? Hier ist die Klassenbalance extrem (0,1% Betrug), was Accuracy als Metrik unbrauchbar macht und die ROC-AUC umso wichtiger.

Warum ist die logistische Regression trotz moderner Machine-Learning-Methoden so beliebt? Drei Gründe: Sie ist *interpretierbar* (Odds Ratios erklären den Einfluss jeder Variable), sie ist *schnell* (auch auf Millionen von Datenpunkten in Sekunden geschätzt), und sie dient als *robuste Baseline* – jedes komplexere Modell sollte mindestens so gut sein wie die logistische Regression, sonst lohnt sich die Komplexität nicht.

Joseph Berkson erfindet "Logit" an der Mayo Clinic (1944)

Die logistische Funktion war bereits seit dem 19. Jahrhundert bekannt – der belgische Mathematiker Pierre-François Verhulst verwendete sie 1838 zur Modellierung von Bevölkerungswachstum. Aber die Verwendung als statistisches Modell für binäre Daten geht auf Joseph Berkson zurück, einen Statistiker und Arzt an der Mayo Clinic in Rochester, Minnesota.

Berkson suchte 1944 nach einer praktikablen Alternative zum Probit-Modell, das Chester Bliss 1934 eingeführt hatte. Das Probit-Modell basierte auf der Normalverteilungs-CDF und war mathematisch schwer handhabbar. Berksons logistisches Modell war rechnerisch einfacher und lieferte fast identische Ergebnisse. Er prägte den Namen "Logit" als Kurzform aus "log" und "unit" – analog zu "Probit" (= "probability unit").

Heute ist die logistische Regression das weltweit meistgenutzte Klassifikationsmodell. In der Medizin, in den Sozialwissenschaften und in der Wirtschaft ist sie der Standard für die Analyse binärer Outcomes. Was Berkson für Patienten an der Mayo Clinic entwickelte, nutzt Lena 80 Jahre später für Kundendaten bei DataCo.

Häufige Fehler

Irrtum 1: "Logistische Regression ist keine Regression." Doch, sie ist eine Regression – auf der Log-Odds-Skala. Die Koeffizienten beschreiben eine lineare Beziehung zwischen den Prädiktoren und den Log-Odds. Der Name "Regression" ist korrekt, auch wenn die Zielvariable binär ist. Die Verwechslung entsteht, weil das *Ergebnis* eine Klassifikation ist, aber die *Methode* ist eine Regression.

Irrtum 2: "Accuracy ist die beste Metrik für Klassifikation." Irreführend, besonders bei unbalancierten Klassen. Wenn nur 5% der Kunden kündigen, erreicht ein Modell, das immer "Bleibt" vorhersagt, 95% Accuracy – ohne auch nur einen einzigen Kündiger zu erkennen. Precision, Recall, $F1$ und AUC-ROC sind fast immer aussagekräftiger.

Irrtum 3: "Odds Ratio = Wahrscheinlichkeitsverhältnis." Nein. $OR = 2{,}5$ bedeutet *nicht*, dass die Wahrscheinlichkeit 2,5-mal so hoch ist. Die Odds Ratio operiert auf Odds, nicht auf Wahrscheinlichkeiten. Bei seltenen Ereignissen ($p < 0{,}10$) ist die Näherung $OR \approx$ Relatives Risiko vertretbar, bei häufigen Ereignissen überschätzt die OR den tatsächlichen Wahrscheinlichkeitsunterschied erheblich.

8.10 R-Code: glm, Odds Ratios, und ROC

R-Code

```r
# --- Logistische Regression bei DataCo ---

library(tidyverse)

# Modell schaetzen
logit_model <- glm(
  churn ~ premium + alter + nutzung_monate + beschwerden,
  data = dataco,
  family = binomial
)

# Zusammenfassung
summary(logit_model)

# Odds Ratios mit 95%-Konfidenzintervall
exp(cbind(OR = coef(logit_model), confint(logit_model)))

# Vorhersagen als Wahrscheinlichkeiten
dataco$pred_prob <- predict(logit_model, type = "response")

# Confusion Matrix mit caret
library(caret)
dataco$pred_class <- ifelse(dataco$pred_prob > 0.5, 1, 0)
confusionMatrix(
  factor(dataco$pred_class),
  factor(dataco$churn),
  positive = "1"
)

# ROC-Kurve und AUC mit pROC
library(pROC)
roc_obj <- roc(dataco$churn, dataco$pred_prob)
plot(roc_obj, main = "ROC-Kurve: DataCo Churn-Modell")
auc(roc_obj)
# Lenas Ergebnis: AUC = 0.82
```

8.11 Übungsaufgaben

Aufgabe 8.1 – OLS vs. Logit

Erklären Sie in eigenen Worten, warum eine lineare Regression für binäre Zielvariablen problematisch ist. Nennen Sie drei konkrete Probleme und erklären Sie, wie die logistische Regression jedes davon löst.

Aufgabe 8.2 – Odds Ratio berechnen

In einer logistischen Regression ist der Koeffizient für die Variable "Beschwerden" $\beta = 0{,}47$. Berechnen Sie die Odds Ratio. Ein Kunde hat aktuell eine Churn-Wahrscheinlichkeit von $p = 0{,}30$. Wie hoch ist seine Churn-Wahrscheinlichkeit nach einer zusätzlichen Beschwerde? Rechnen Sie über den Weg: $p \to$ Odds $\to$ neue Odds $\to$ neues p.

Aufgabe 8.3 – Confusion Matrix interpretieren

Ein Modell zur Betrugserkennung bei $n = 10.000$ Transaktionen (davon 50 Betrug) liefert folgende Confusion Matrix:

	Pred. Kein Betrug	Pred. Betrug
Kein Betrug	9920	30
Betrug	10	40

Berechnen Sie Accuracy, Precision, Recall und $F1$. Warum ist Accuracy hier irreführend? Welche Metrik würde die Bank bevorzugen, und warum?

Aufgabe 8.4 – AUC interpretieren

Zwei Modelle für Churn-Prediction haben AUC-Werte von 0,72 und 0,88. Erklären Sie in eigenen Worten, was der Unterschied praktisch bedeutet. Wie würden Sie entscheiden, ob die Verbesserung von 0,72 auf 0,88 den Einsatz des komplexeren Modells rechtfertigt?

Aufgabe 8.5 – Vollständige Modellierung

Sie erhalten Daten zu 1000 Kreditanträgen mit den Features Einkommen, Alter, bestehende Schulden und Anstellungsdauer. Die Zielvariable ist "Kreditausfall" (1 = Ausfall, 0 = kein Ausfall, Basisrate: 8%). Beschreiben Sie Ihren vollständigen Modellierungsprozess: Welches Modell wählen Sie? Welche Metriken berichten Sie und warum? Warum ist Accuracy hier keine gute Wahl?

8.12 Lenas Reise: Zusammenfassung

Vor acht Kapitel stand Lena vor einer Tabellenkalkulation mit 50.000 Zeilen und dem Auftrag ihres Chefs: "Sag mir, wer kündigt." Was auf den ersten Blick nach einer einfachen Vorhersageaufgabe klang, entpuppte sich als eine Reise durch die Grundlagen der statistischen Modellierung.

Lena begann mit der Frage, ob sie ihren OLS-Schätzern vertrauen kann – und lernte, dass BLUE nur unter bestimmten Annahmen gilt. Sie entdeckte, dass korrelierte Prädiktoren ihre Koeffizienten instabil machen (Multikollinearität), dass ein perfekt angepasstes Modell auf neuen Daten versagen kann (Overfitting), und dass Regularisierung und Kreuzvalidierung die Werkzeuge sind, um robuste Modelle zu bauen. Sie erkannte, dass Korrelation nicht Kausalität bedeutet – und dass nur ein randomisiertes Experiment (A/B-Test) kausale Schlussfolgerungen zulässt. Und schließlich lernte sie, dass binäre Outcomes ein eigenes Modell brauchen: die logistische Regression, die Wahrscheinlichkeiten zwischen 0 und 1 garantiert und durch Odds Ratios interpretierbar bleibt.

Lenas Churn-Modell bei DataCo erreicht eine AUC von 0,82. Ihr A/B-Test hat bewiesen, dass das neue Onboarding die Retention kausal um 8 Prozentpunkte erhöht. Sie kann dem Board nicht nur sagen, *wer* wahrscheinlich kündigt, sondern auch, *welche Maßnahme* dagegen wirkt – und das mit einem Modell, dessen Ergebnisse sie Zeile für Zeile erklären kann. Von der deskriptiven Analyse über die Inferenz bis zur kausalen Schlussfolgerung: Das ist der Weg von der Datenanalyse zur Data Science.

Kernaussage: Logistische Regression ist die Basis für Klassifikation. Auf der Log-Odds-Skala ist das Modell linear, und e^{β} liefert interpretierbare Odds Ratios. Bei unbalancierten Klassen sind Precision, Recall und AUC-ROC aussagekräftiger als Accuracy.

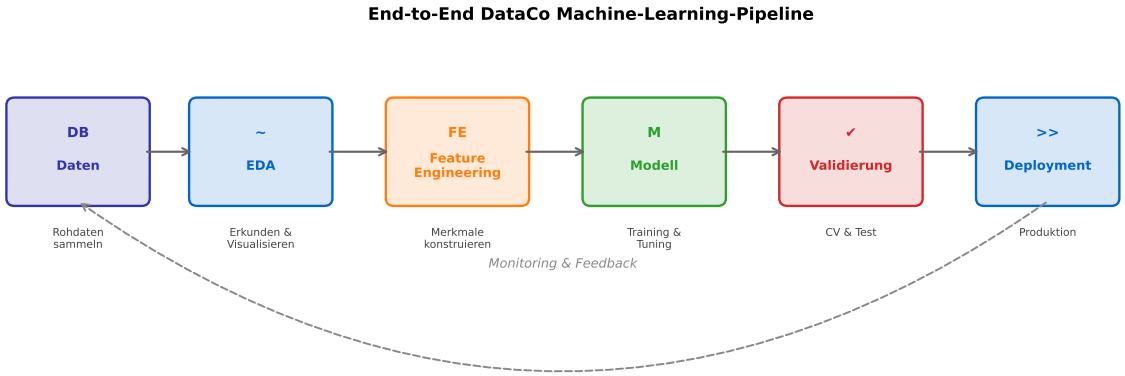


Abbildung 21: Die vollständige Data-Science-Pipeline: Von den Rohdaten zur Entscheidung

Lösungen zu den Übungsaufgaben

Abschnitt 1: BLUE und Gauss-Markov

Lösung 1.1: (a) Mittelwert: $\bar{x} = (98{+}102{+}\cdots{+}100)/10 = 1000/10 = 100{,}0\,\text{g}$. Der Mittelwert trifft den wahren Wert exakt, die Waage ist erwartungstreu. (b) Die Abweichungen vom Mittelwert sind $-2, +2, -1, +1, 0, +3, -3, +1, -1, 0$.

$$s^2 = \frac{4{+}4{+}1{+}1{+}0{+}9{+}9{+}1{+}1{+}0}{9} = \frac{30}{9} \approx 3{,}33$$

Die Standardabweichung ist $s \approx 1{,}83\,\text{g}$. Die Stabilität ist gut – die Messungen schwanken nur um ca. $\pm 2\,\text{g}$. (c) Eigenschaft (a) entspricht **Unbiased** (U in BLUE): Im Mittel wird der wahre Wert getroffen. Eigenschaft (b) entspricht **Best** (B in BLUE): Geringe Varianz, also hohe Präzision.

Lösung 1.2: (a) **Annahme 1 (Linearität in den Parametern)** ist verletzt – der exponentielle Zusammenhang kann nicht durch ein lineares Modell abgebildet werden. (b) **Annahme 2 (Zufällige Stichprobe / Unabhängigkeit)** ist verletzt – aufeinanderfolgende Beobachtungen in einer Zeitreihe sind abhängig (Autokorrelation). (c) **Annahme 3 (Keine perfekte Multikollinearität)** ist verletzt – Jahreskosten = 12× Monatliche Kosten ist eine exakte Linearkombination. (d) **Annahme 5 (Homoskedastizität)** ist verletzt – die zunehmende Streuung bei größerem Umsatz ist ein Trichtermuster (Heteroskedastizität).

Lösung 1.3: (a) $t = \hat{\beta}_1 / SE = 3{,}5/0{,}7 = 5{,}0$. Der kritische Wert bei $\alpha = 0{,}05$ (zweiseitig) ist ca. 1,97 (bei $n = 200$, nahe 1,96). Da $5{,}0 > 1{,}97$, ist der Koeffizient hochsignifikant. (b) $t = 3{,}5/2{,}1 \approx 1{,}67$. (c) Da $1{,}67 < 1{,}97$, ist $\hat{\beta}_1$ *nicht* mehr signifikant. Der Koeffizient selbst hat sich nicht geändert, aber der aufgeblähte Standardfehler führt dazu, dass die Signifikanz verschwindet. Lena könnte einen echten Effekt übersehen, weil die Inferenz durch Heteroskedastizität verzerrt ist. Robuste Standardfehler (z. B. HC1) wären die Lösung.

Lösung 1.4: (a) **Annahme 1 (Linearität)** ist verletzt. Ein U-förmiges Muster im Residuen-vs-Fitted-Plot zeigt, dass der wahre Zusammenhang nichtlinear ist. (b) Die Koeffizienten sind *verzerrt* (biased), weil das Modell den wahren funktionalen Zusammenhang nicht abbildet. OLS ist nicht mehr erwartungstreu. (c) Zwei Lösungen: (1) Polynomiale Terme aufnehmen (z. B. x^2), um die Krümmung zu modellieren. (2) Eine Transformation der Variablen (z. B. $\log(x)$) verwenden, um den Zusammenhang zu linearisieren.

Lösung 1.5: (a) Ja, $\hat{\beta}_1^{\text{OLS}}$ ist BLUE, denn per Definition gilt das Gauss-Markov-Theorem bei erfüllten Annahmen: OLS ist Best Linear Unbiased Estimator. (b) Nein, der Median ist *nicht* BLUE. Er ist zwar robust, aber er ist kein *linearer* Schätzer (er lässt sich nicht als gewichtete Summe $\sum w_i y_i$ darstellen). Die L-Eigenschaft ist verletzt. (c) Nein, Ridge ist *nicht* BLUE. Ridge ist zwar linear, aber *nicht* erwartungstreu – der Strafterm führt bewusst eine Verzerrung ein ($E[\hat{\beta}^{\text{Ridge}}] \neq \beta$). Die U-Eigenschaft ist verletzt. Dafür kann Ridge einen niedrigeren Gesamtfehler (MSE) haben, weil die Varianzreduktion die Verzerrung überkompensiert.

Abschnitt 2: Multikollinearität und VIF

Lösung 2.1: (a) $VIF_1 = \frac{1}{1-R_1^2} = \frac{1}{1-0{,}75} = \frac{1}{0{,}25} = 4{,}0$. (b) Der Standardfehler ist um den Faktor $\sqrt{VIF_1} = \sqrt{4{,}0} = 2{,}0$ aufgeblasen, also doppelt so groß wie ohne Multikollinearität. (c) Nach den Faustregeln ist $VIF = 4{,}0 < 5$ noch unkritisch, liegt aber nahe an der Grenze. Es lohnt sich, die Situation im Auge zu behalten.

Lösung 2.2: (a) **Symptom 1: Hohes R^2, aber keine signifikanten Koeffizienten.** Das Gesamtmodell erklärt fast so viel Varianz wie vorher (R^2 steigt von 0,35 auf 0,38), aber die neuen Prädiktoren führen zu Multikollinearität, sodass kein einzelner Koeffizient mehr signifikant wird. (b) R^2 steigt trotzdem, weil die hinzugefügten Variablen zumindest etwas zusätzliche Varianz erklären. R^2 kann durch Hinzufügen von Prädiktoren nie sinken – es kann nur gleich bleiben

oder steigen. Die Multikollinearität betrifft die *einzelnen* Koeffizienten, nicht die Gesamterklärungskraft. (c) Lena sollte die **VIF-Werte** berechnen (`car::vif(model)`), um zu quantifizieren, welche Prädiktoren die Multikollinearität verursachen.

Lösung 2.3: (a) Um einzelne Einflussfaktoren zu identifizieren, muss Lena die Multikollinearität behandeln. Empfehlung: Entweder eine der korrelierten Variablen entfernen (basierend auf inhaltlicher Überlegung, nicht nur auf dem VIF) oder **Ridge-Regression** verwenden, die die Koeffizienten stabilisiert. (b) Nein, für reine Vorhersage muss sie nicht unbedingt handeln. Multikollinearität schadet der *Vorhersagegenauigkeit* des Gesamtmodells kaum – das R^2 ist fast gleich. Nur wenn sie einzelne Koeffizienten interpretieren will, ist Handlung nötig. (c) Wenn "monatliche Kosten" kausal auf Churn wirkt und "Vertragslaufzeit" nur ein korrelierter Proxy ist, würde das Entfernen der Kosten-Variable den kausalen Mechanismus aus dem Modell nehmen. Die Entscheidung, welche Variable entfernt wird, muss auf Fachwissen und kausaler Überlegung basieren.

Lösung 2.4: (a) Das Paar X_1 und X_2 mit $r = 0{,}95$ ist offensichtlich problematisch – eine Korrelation nahe 1. (b) $VIF_1 \approx \frac{1}{1-0{,}90} = \frac{1}{0{,}10} = 10{,}0$. Das ist nach den Faustregeln schwer problematisch. (c) Der wahre VIF ist vermutlich höher, weil die Approximation $R_1^2 \approx r_{12}^2$ nur die paarweise Korrelation mit X_2 berücksichtigt. Der echte R_1^2 stammt aus einer multiplen Regression von X_1 auf X_2 *und* X_3 gleichzeitig. Da X_3 mit X_1 und X_2 auch korreliert ist ($r_{13} = 0{,}20$, $r_{23} = 0{,}30$), wird $R_1^2 > r_{12}^2$ sein.

Lösung 2.5: (a) Multikollinearität blaest die Standardfehler einzelner Koeffizienten auf, weil das Modell nicht zuverlässig aufteilen kann, welcher Anteil des Effekts auf welchen Prädiktor zurückgeht. Dadurch werden die Koeffizienten instabil und schwer interpretierbar. (b) Für die Vorhersage nutzt das Modell die *gemeinsame* Information aller Prädiktoren. Solange die Korrelationsstruktur in neuen Daten ähnlich bleibt wie in den Trainingsdaten, funktioniert die Vorhersage trotzdem gut. Die einzelnen Koeffizienten sind zwar instabil, aber ihre Kombination ergibt trotzdem gute Vorhersagen. (c) Für die **Interpretation** ist es wichtiger, die Multikollinearität zu behandeln. Wer einzelne Koeffizienten interpretieren und daraus Handlungsempfehlungen ableiten will (z. B. "Welcher Faktor treibt Churn am stärksten?"), braucht stabile, zuverlässige Schätzungen.

Abschnitt 3: Bias-Varianz-Tradeoff

Lösung 3.1: Das Problem ist **Overfitting**: Die große Lücke zwischen Trainings-R^2 (0,98) und Test-R^2 (0,41) – also $0{,}98 - 0{,}41 = 0{,}57$ – zeigt, dass das Modell das Rauschen in den Trainingsdaten gespeichert hat statt das Muster zu erkennen. Drei Maßnahmen: (1) **Modellkomplexität reduzieren** – weniger Features oder ein einfacheres Modell wählen. (2) **Regularisierung** einsetzen (Ridge oder Lasso), um die Koeffizienten zu schrumpfen und die Varianz zu reduzieren. (3) **Kreuzvalidierung** verwenden statt eines einzelnen Train/Test-Splits, um die Out-of-Sample-Performance robuster zu schätzen.

Lösung 3.2: (a) Gesamtfehler $= \text{Bias}^2 + \text{Varianz} + \sigma^2 = 0{,}6^2 + 0{,}1 + 0{,}3^2 = 0{,}36 + 0{,}10 + 0{,}09 = 0{,}55$. (b) Der **Bias**^{2} dominiert mit 0,36 von 0,55 (65% des Gesamtfehlers). Das Modell vereinfacht den wahren Zusammenhang zu stark. (c) Lena sollte das Modell **komplexer** machen (z. B. mehr Features aufnehmen oder ein flexibleres Modell wählen), um den Bias zu reduzieren. Die Varianz ist mit 0,1 vergleichsweise gering, es gibt also Spielraum für mehr Komplexität.

Lösung 3.3: (a) Lineare Regression auf quadratischen Zusammenhang (viele Daten): **Hoher Bias, niedrige Varianz**. Das Modell kann die Krümmung nicht abbilden (systematischer Fehler), aber viele Daten sorgen für Stabilität. (b) Polynom Grad 20 auf 15 Datenpunkte: **Niedriger Bias, hohe Varianz** (bzw. bei extremem Overfitting auch hoher Bias möglich). Mit mehr Parametern (20) als Datenpunkten (15) ist das Modell maßiv überparametrisiert und extrem instabil. (c) Polynom Grad 3 auf quadratischen Zusammenhang (viele Daten): **Niedriger Bias, niedrige Varianz** – der ideale Fall. Das Modell hat genug Flexibilität für die quadratische Struktur und genug Daten für Stabilität. (d) Konstantes Modell $\hat{y} = \bar{y}$ mit 5 Datenpunkten: **Hoher Bias,**

hohe Varianz. Das Modell ignoriert alle Prädiktoren (hoher Bias), und bei nur 5 Datenpunkten schwankt selbst der Mittelwert stark (hohe Varianz).

Lösung 3.4: Die Faustregel verlangt 10–20 Beobachtungen pro Prädiktor. Bei $n = 300$ und der konservativen Grenze ($n/p \geq 20$): $p_{\max} = 300/20 = 15$ Prädiktoren. Bei der liberalen Grenze ($n/p \geq 10$): $p_{\max} = 300/10 = 30$ Prädiktoren. Wenn der Chef 80 Features verlangt, ergibt sich $n/p = 300/80 = 3{,}75$ – weit unter der kritischen Schwelle. Overfitting ist quasi garantiert. Lena müsste Regularisierung (Lasso/Ridge) oder Feature Selection einsetzen, um die effektive Zahl der Parameter zu reduzieren.

Lösung 3.5: (a) Beim **Netflix Prize** spielte **Overfitting** die Hauptrolle: Das Gewinnermodell war ein Ensemble aus über 100 Modellen, das die Trainingsdaten zu gut anpasste und in der Praxis zu komplex war. Bei der **Challenger-Katastrophe** spielte **Selection Bias** die Hauptrolle: Die Ingenieure analysierten nur Flüge *mit* Schäden und ignorierten Flüge ohne Schäden, wodurch der Zusammenhang zwischen Temperatur und O-Ring-Versagen unsichtbar wurde. (b) Bei Netflix hätte ein **einfacheres Modell** (weniger Ensemble-Komponenten) oder striktere Regularisierung geholfen. Bei Challenger hätte die **Einbeziehung aller Datenpunkte** (auch Flüge ohne Schäden) den Zusammenhang sofort sichtbar gemacht.

Abschnitt 4: Regularisierung

Lösung 4.1: Die L1-Strafe (Lasso) definiert einen Constraint-Bereich in Form eines **Diamanten** (Raute), die L2-Strafe (Ridge) einen **Kreis**. Die OLS-Lösung liegt typischerweise außerhalb des Constraint-Bereichs. Die regularisierte Lösung ist der Punkt, an dem die elliptischen Konturlinien der Fehlerfunktion den Constraint-Bereich zuerst beruehren. Beim Diamanten beruehren die Ellipsen den Constraint-Bereich am ehesten an einer *Ecke* – und die Ecken des Diamanten liegen auf den Achsen, wo mindestens ein Koeffizient exakt Null ist. Beim Kreis beruehren die Ellipsen den Rand typischerweise an einem Punkt, der nicht auf einer Achse liegt – daher werden Koeffizienten nur geschrumpft, aber nicht auf exakt Null gesetzt.

Lösung 4.2: Bei $\lambda = 0$: Alle Koeffizienten entsprechen der OLS-Lösung. Bei steigendem λ: Die Koeffizienten werden zunehmend Richtung Null geschrumpft. Dabei verschwinden zuerst die Features mit dem *schwächsten* Einfluss auf die Zielvariable (kleinster $|\beta|$). Das Feature, das am laengsten "überlebt" (als letztes Null wird), ist dasjenige mit dem stärksten Zusammenhang zur Zielvariable – es trägt die meiste praediktive Information. Bei sehr großem λ: Alle Koeffizienten sind Null, das Modell reduziert sich auf den Intercept $\hat{y} = \bar{y}$.

Lösung 4.3: (a) **Ridge**: Mit nur 8 Features, die alle theoretisch begründet sind, soll kein Feature eliminiert werden. Ridge schrumpft alle Koeffizienten proportional und stabilisiert die Schätzung. (b) **Lasso**: Bei 200 Features, von denen die meisten vermutlich irrelevant sind, ist Feature Selection entscheidend. Lasso kann die irrelevanten Features automatisch auf Null setzen und ein sparsames Modell liefern. (c) **Elastic Net**: Mit 5 Gruppen korrelierter Features hat Lasso ein Problem – es wähltaus jeder Gruppe nur ein Feature und ignoriert die anderen. Elastic Net kombiniert die L1-Selektion mit der L2-Gruppierung: Korrelierte Features werden tendenziell gemeinsam behalten oder gemeinsam entfernt.

Lösung 4.4: `lambda.min` ist der λ-Wert, bei dem der CV-Fehler sein *Minimum* erreicht – das Modell mit der besten Vorhersageleistung. `lambda.1se` ist der größte λ-Wert, dessen CV-Fehler noch innerhalb einer Standardabweichung des Minimums liegt – das sparsamste Modell, das fast genauso gut ist. Lena bevorzugt `lambda.1se`, wenn sie ein einfaches, interpretierbares Modell will (z. B. für die Board-Präsentation). Sie bevorzugt `lambda.min`, wenn maximale Vorhersagegenauigkeit im Vordergrund steht (z. B. für ein automatisiertes Scoring-System).

Lösung 4.5: (a) `lambda.min = 0,023`: Bei diesem Wert ist der CV-Fehler am kleinsten; 9 von 12 Features werden verwendet. `lambda.1se = 0,089`: Bei diesem größeren λ ist der CV-Fehler nur unwesentlich schlechter (innerhalb einer Standardabweichung); nur 5 von 12 Features werden benötigt. (b) `lambda.1se`, da es das sparsamere Modell mit nur 5 Features liefert – einfacher zu erklären und zu kommunizieren. (c) Der Preis ist ein etwas höherer CV-Fehler (innerhalb einer SE

vom Minimum). Das Modell könnte in manchen Situationen etwas weniger genau vorhersagen, weil es auf 4 Features verzichtet, die einen kleinen Beitrag leisten.

Abschnitt 5: Kreuzvalidierung

Lösung 5.1: (a) Jeder Fold enthält $n/K = 120/10 = 12$ Datenpunkte. (b) In jedem Durchlauf werden $n - n/K = 120 - 12 = 108$ Datenpunkte zum Training verwendet. (c) Jeder Datenpunkt ist in genau $K - 1 = 9$ der 10 Durchläufe im Trainingsset enthalten. Er wird also insgesamt 9-mal zum Training herangezogen und genau 1-mal zur Validierung.

Lösung 5.2: LOOCV (Leave-One-Out CV) hat den **geringsten Bias**, weil fast alle Daten $(n - 1)$ zum Training verwendet werden. Allerdings hat LOOCV zwei Nachteile: (1) **Hohe Varianz**: Da sich die n Trainingssets nur um eine Beobachtung unterscheiden, sind die Modelle stark korreliert – die gemittelten Fehlerschätzungen schwanken wenig, können aber instabil sein. (2) **Hoher Rechenaufwand**: Das Modell muss n-mal trainiert werden. Bei 10-Fold CV muss es nur 10-mal trainiert werden, und die 10 Trainingssets überlappen weniger, was zu einer stabileren (weniger varianzreichen) Fehlerschätzung führt. In der Praxis ist 10-Fold CV deshalb der Standard.

Lösung 5.3: (a) Bei $K = 5$ ohne Stratifizierung könnte ein Fold rein zufällig nur 5% Kündiger enthalten und ein anderer 25%. Das Modell würde auf stark unterschiedlichen Verteilungen trainiert und validiert, was zu instabilen CV-Ergebnissen führt. Im Extremfall könnte ein Fold sogar gar keine Kündiger enthalten. (b) Stratified K-Fold stellt sicher, dass jeder Fold *denselben Anteil* an Kündigern enthält wie der Gesamtdatensatz (hier 15%). Die Aufteilung in Folds erfolgt innerhalb jeder Klasse separat, sodass die Klassenverteilung in jedem Fold repräsentativ ist.

Lösung 5.4: Wenn man λ per CV wähltund die Performance auf denselben CV-Daten bewertet, entsteht ein **optimistischer Bias**: Das Modell wurde auf die spezifischen Daten optimiert, und die Performance-Schätzung spiegelt diese Optimierung wider. Es ist, als würde man Prüfungsfragen zum Lernen verwenden und dann dieselben Fragen als Test stellen. Bei **Nested CV** gibt es eine *äußere* und eine *innere* Schleife. Die innere Schleife wähltdie optimalen Hyperparameter (z. B. λ) per CV. Die äußere Schleife bewertet die Performance auf Daten, die weder zum Training noch zur Hyperparameter-Wahl verwendet wurden. So wird die Performance-Schätzung fair und unverzerrt.

Lösung 5.5: Modell A hat eine höhere mittlere Accuracy (0,80), aber auch eine größere Streuung ($SD = 0{,}08$). Modell B hat eine etwas niedrigere mittlere Accuracy (0,77), ist aber deutlich stabiler ($SD = 0{,}02$). Empfehlung: **Modell B**. Der Unterschied in der mittleren Accuracy beträgt nur 3 Prozentpunkte, liegt aber innerhalb der Standardabweichung von Modell A ($0{,}80 \pm 0{,}08$). Die hohe Streuung von Modell A deutet darauf hin, dass seine Performance stark vom konkreten Datensplit abhängt – es ist instabil und könnte auf neuen Daten ähnlich schwanken. Modell B liefert zuverlässig $0{,}77 \pm 0{,}02$ und ist deshalb vertrauenswuerdiger.

Abschnitt 6: Kausalität

Lösung 6.1: (a) Wahrscheinlich **Scheinkorrelation**. Confounder: Sozioökonomischer Status der Familie. Kinder aus wohlhabenderen Familien frühstücken häufiger *und* haben bessere Lernbedingungen. (b) **Scheinkorrelation**. Confounder: Ländlichkeit/Urbanisierung. Ländliche Regionen haben sowohl mehr Stöche (Brutgebiete) als auch höhere Geburtenraten. (c) **Möglicherweise kausal**, wenn aus einem randomisierten Experiment (RCT). Bei einer Beobachtungsstudie könnte ein Confounder vorliegen: Patienten, die Medikament X nehmen, gehen regelmäßiger zum Arzt und ändern ihren Lebensstil. (d) **Scheinkorrelation**. Confounder: Stadtgröße. Größere Städte haben sowohl mehr Feuerwehrleute als auch mehr Brände.

Lösung 6.2: (a) **Stufe 1 (Sehen/Beobachten)**: Das ist eine reine Korrelationsaussage – wer Newsletter liest, kauft mehr (Assoziation). (b) **Stufe 2 (Tun/Intervention)**: Das ist eine kausale Interventionsfrage – was passiert, *wenn* wir aktiv eingreifen? (c) **Stufe 3 (Vorstellen/**

Kontrafaktisch): Das ist eine kontrafaktische Frage über einen einzelnen Kunden – was *wäre gewesen, wenn*? (d) **Stufe 1 (Sehen/Beobachten)**: Das ist eine beobachtete Assoziation aus epidemiologischen Daten (keine experimentelle Intervention).

Lösung 6.3: (a) Knoten: Homeoffice, Produktivität, Berufserfahrung, Aufgabentyp. (b) Pfeile: Berufserfahrung → Homeoffice (erfahrene Mitarbeiter dürfen eher ins Homeoffice), Berufserfahrung → Produktivität (Erfahrung steigert Produktivität), Aufgabentyp → Homeoffice (kreative Aufgaben werden öfter remote erledigt), Aufgabentyp → Produktivität (Aufgabentyp beeinflusst Produktivität), und Homeoffice → Produktivität (der kausale Effekt, den wir schätzen wollen). (c) Das Adjustment Set ist {Berufserfahrung, Aufgabentyp}. Man muss für beide Confounders kontrollieren, um den kausalen Effekt von Homeoffice auf Produktivität unverzerrt zu schätzen.

Lösung 6.4: (a) **Wettbewerberaktivität** hat den größeren Effekt. Die Kontrolle für Saison reduziert die Korrelation nur von 0,60 auf 0,55 (Reduktion um 0,05). Die zusätzliche Kontrolle für Wettbewerberaktivität reduziert sie von 0,55 auf 0,15 (Reduktion um 0,40). (b) Nein, Lena kann das nicht schliessen. Die nicht-signifikante partielle Korrelation nach Kontrolle für zwei Variablen bedeutet nur, dass in diesem Modell kein signifikanter Zusammenhang mehr besteht. Es könnte aber sein, dass (i) der Stichprobenumfang zu klein ist, (ii) die Operationalisierung ungenau ist, oder (iii) der kausale Pfad über Wettbewerberaktivität läuft (Mediation statt Confounding). (c) Damit die partielle Korrelation den kausalen Effekt schätzt, müsste gelten, dass *alle* relevanten Confounders kontrolliert wurden und kein Collider-Bias vorliegt. Diese Annahme ist in Beobachtungsstudien nie sicher erfüllt.

Lösung 6.5: (a) Ein Vorher-Nachher-Vergleich reicht nicht, weil andere Faktoren sich gleichzeitig geändert haben könnten (Saison, Wettbewerb, allgemeine Markttrends). Ohne Kontrollgruppe kann jede beobachtete Änderung ein Confounder sein. (b) **Design**: 1000 Kunden werden per Zufall in zwei Gruppen geteilt. Kontrollgruppe ($n = 500$): kein Treueprogramm. Treatment-Gruppe ($n = 500$): erhält das neue Treueprogramm. Primäre Metrik: 90-Tage-Kündigungsrate. Potenzielle Confounders (die durch die Randomisierung kontrolliert werden): Alter, Premium-Status, bisherige Nutzungsdauer, Zufriedenheit. (c) Stufe 2 (Tun): Ein randomisiertes Experiment erlaubt kausale Schlussfolgerungen über den *durchschnittlichen* Behandlungseffekt (ATE).

Abschnitt 7: A/B-Testing

Lösung 7.1: Gegeben: $d = 0{,}4$, $\alpha = 0{,}05$ ($z_{\alpha/2} = 1{,}96$), Power $= 0{,}80$ ($z_\beta = 0{,}84$).

$$n \approx \frac{2\,(z_{\alpha/2} + z_\beta)^2}{d^2} = \frac{2 \times (1{,}96 + 0{,}84)^2}{0{,}4^2} = \frac{2 \times (2{,}80)^2}{0{,}16} = \frac{2 \times 7{,}84}{0{,}16} = \frac{15{,}68}{0{,}16} = 98$$

Man braucht **mindestens 98 Besucher pro Gruppe**, also insgesamt 196. (Aufgerundet: $n = 99$ pro Gruppe, falls man konservativ rechnet.)

Lösung 7.2: Gepoolte Standardabweichung:

$$s_p = \sqrt{\frac{(n_T - 1)\,s_T^2 + (n_C - 1)\,s_C^2}{n_T + n_C - 2}} = \sqrt{\frac{199 \times 14^2 + 199 \times 12^2}{398}}$$

$$= \sqrt{\frac{199 \times 196 + 199 \times 144}{398}} = \sqrt{\frac{39\,004 + 28\,656}{398}} = \sqrt{\frac{67\,660}{398}} = \sqrt{170{,}0} \approx 13{,}04$$

Cohens d:

$$d = \frac{\bar{X}_T - \bar{X}_C}{s_p} = \frac{52 - 48}{13{,}04} = \frac{4}{13{,}04} \approx 0{,}31$$

Nach Cohens Daumenregel ist $d = 0{,}31$ ein **kleiner bis mittlerer Effekt** (zwischen 0,2 und 0,5).

Lösung 7.3: Das Ergebnis ist nicht verlässlich, weil der Produktmanager **Peeking** betrieben hat. Er hat während des laufenden Tests wiederholt auf Signifikanz getestet. Bei $k = 8$

Zwischenanalysen beträgt die ungefähre effektive Falsch-Positiv-Rate:

$$1 - 0{,}95^8 = 1 - 0{,}6634 \approx 0{,}337$$

Die Wahrscheinlichkeit eines falsch-positiven Ergebnisses liegt also bei ca. **33,7%** statt bei den angestrebten 5%. Das beobachtete $p = 0{,}04$ könnte leicht ein Zufallsbefund sein. Die Lösung: Stichprobengröße vorher festlegen und erst am geplanten Ende (30 Tage) auswerten, oder ein sequentielles Testverfahren verwenden.

Lösung 7.4: Die Wahrscheinlichkeit, mindestens ein falsch-positives Ergebnis zu finden:

$$1 - (1 - 0{,}05)^{15} = 1 - 0{,}95^{15} \approx 1 - 0{,}4633 \approx 0{,}537$$

Es besteht eine **53,7%-ige Wahrscheinlichkeit**, mindestens ein falsch-positives Ergebnis zu finden – mehr als jedes zweite Mal! Die Bonferroni-Korrektur setzt das Signifikanzniveau pro Test auf:

$$\alpha_{\text{korr}} = \frac{0{,}05}{15} \approx 0{,}0033$$

Jede einzelne Metrik müsste also auf dem 0,33%-Niveau signifikant sein.

Lösung 7.5: Schritt 1 – Hypothese: H_0: Die neue E-Mail-Kampagne hat keinen Effekt auf die Klickrate. H_1: Die neue Kampagne erhöht die Klickrate. **Schritt 2 – Primäre Metrik**: Klickrate (Anteil der Empfänger, die auf den Link klicken), gemessen 7 Tage nach Versand. **Schritt 3 – MDE**: $d = 0{,}25$ (kleiner Effekt). Dies ist der kleinste Effekt, der geschäftlich noch relevant ist. **Schritt 4 – Sample Size**: $n \approx \frac{2\times(1{,}96+0{,}84)^2}{0{,}25^2} = \frac{2\times 7{,}84}{0{,}0625} = \frac{15{,}68}{0{,}0625} \approx 251$ pro Gruppe, also insgesamt ca. 502 Empfänger. **Schritt 5 – Laufzeit**: 7 Tage für die Klickmessung plus 2–3 Tage Puffer, also ca. 10 Tage. Versand an einem typischen Wochentag starten, um Saisonalitätseffekte zu vermeiden. Alle Entscheidungen werden vorab dokumentiert (Pre-Registration).

Abschnitt 8: Logistische Regression

Lösung 8.1: Drei Probleme der linearen Regression bei binären Zielvariablen und deren Lösung durch logistische Regression: (1) **Vorhersagen außerhalb** $[0, 1]$: OLS kann negative Wahrscheinlichkeiten oder Werte > 1 liefern. Die logistische Funktion $P = 1/(1 + e^{-z})$ bildet jeden Wert in das Intervall $(0, 1)$ ab. (2) **Nichtlinearer Zusammenhang**: Der wahre Zusammenhang zwischen X und $P(Y = 1)$ ist S-förmig, nicht linear. Die Sigmoid-Funktion modelliert genau diese S-Form. (3) **Heteroskedastizität**: Bei binären Daten ist $\text{Var}(Y|X) = P(1 - P)$, also abhängig vom Mittelwert. Logistische Regression verwendet Maximum-Likelihood statt OLS und setzt keine konstante Fehlervarianz voraus.

Lösung 8.2: Odds Ratio: $OR = e^{\beta} = e^{0{,}47} \approx 1{,}60$. Jede zusätzliche Beschwerde multipliziert die Churn-Odds mit dem Faktor 1,60. Schritt-für-Schritt-Berechnung:

- Aktuelle Wahrscheinlichkeit: $p = 0{,}30$.
- Aktuelle Odds: $\frac{p}{1-p} = \frac{0{,}30}{0{,}70} \approx 0{,}4286$.
- Neue Odds: $0{,}4286 \times 1{,}60 \approx 0{,}6857$.
- Neue Wahrscheinlichkeit: $p_{\text{neu}} = \frac{0{,}6857}{1+0{,}6857} = \frac{0{,}6857}{1{,}6857} \approx 0{,}407$.

Die Churn-Wahrscheinlichkeit steigt von 30% auf ca. **40,7%**.

Lösung 8.3: Berechnung der Metriken (TP = 40, FP = 30, TN = 9920, FN = 10):

- Accuracy $= \frac{40+9920}{10\,000} = \frac{9960}{10\,000} = 0{,}996$ (99,6%)
- Precision $= \frac{40}{40+30} = \frac{40}{70} \approx 0{,}571$ (57,1%)
- Recall $= \frac{40}{40+10} = \frac{40}{50} = 0{,}800$ (80,0%)

- $F1 = 2 \times \frac{0{,}571 \times 0{,}800}{0{,}571 + 0{,}800} = 2 \times \frac{0{,}457}{1{,}371} \approx 0{,}667$ (66,7%)

Accuracy ist irreführend, weil ein triviales Modell, das *immer* "Kein Betrug" vorhersagt, bereits $9950/10\,000 = 99{,}5\%$ Accuracy erreichen würde. Die Klassen sind extrem unbalanciert (nur 0,5% Betrug). Die Bank würde **Recall** bevorzugen, weil übersehener Betrug (FN) teurer ist als ein Fehlalarm (FP). Ein Recall von 80% bedeutet, dass 80% der Betrugsfälle erkannt werden – aber 10 von 50 Betrugsfällen bleiben unentdeckt.

Lösung 8.4: Eine AUC von 0,72 bedeutet: In 72% der Fälle rankt das Modell einen zufälligen Kündiger höher als einen zufälligen Bleiber. Eine AUC von 0,88 bedeutet: In 88% der Fälle. Der Unterschied von 16 Prozentpunkten ist erheblich – das bessere Modell trifft deutlich zuverlässigere Unterscheidungen. Ob die Verbesserung den Einsatz rechtfertigt, hängt von den Kosten ab: Wie teuer ist ein übersehener Kündiger (FN)? Wie teuer ist das komplexere Modell (Entwicklung, Wartung, Erklärbarkeit)? Wenn der Customer Lifetime Value hoch ist, lohnt sich die Verbesserung fast immer. Bei niedrigen Margen und hohem Implementierungsaufwand könnte das einfachere Modell ausreichen.

Lösung 8.5: Modellwahl: Logistische Regression, da die Zielvariable binär ist (Ausfall ja/nein). OLS wäre ungeeignet (Vorhersagen außerhalb $[0, 1]$). **Metriken**: Bei einer Basisrate von nur 8% ist Accuracy irreführend – ein triviales Modell, das immer "kein Ausfall" vorhersagt, hätte 92% Accuracy. Stattdessen sollte Lena berichten: (1) **AUC-ROC** als Gesamtmaß für die Trennschärfe, (2) **Precision** (wie viele der als riskant eingestuften Kredite fallen tatsächlich aus?), (3) **Recall** (wie viele der tatsächlichen Ausfälle werden erkannt?), (4) $F1$**-Score** als Kompromiss. **Prozess**: Daten in Trainings- und Testset aufteilen (stratifiziert, um die 8%-Rate zu erhalten), Modell auf Trainingsdaten fitten, Hyperparameter (z. B. Schwellenwert, ggf. Regularisierung) per Kreuzvalidierung wählen, finale Evaluation auf dem Testset. Accuracy ist hier keine gute Wahl, weil sie von der dominanten Klasse (92% Nicht-Ausfälle) dominiert wird und echte Ausfälle kaum gewichtet.